

ARM926EJ-S™-based Microcontroller

AT91SAM9261

Features

- Incorporates the ARM926EJ-S™ ARM® Thumb® Processor
  - DSP Instruction Extensions
  - ARM Jazelle® Technology for Java® Acceleration
  - 16-KByte Data Cache, 16-KByte Instruction Cache, Write Buffer
  - 200 MIPS at 180 MHz
  - Memory Management Unit
  - EmbeddedICE™ In-circuit Emulation, Debug Communication Channel Support
  - Mid-level implementation Embedded Trace Macrocell™
- Additional Embedded Memories
  - 32K Bytes of Internal ROM, Single-cycle Access at Maximum Bus Speed
  - 160K Bytes of Internal SRAM, Single-cycle Access at Maximum Processor or Bus Speed
- External Bus Interface (EBI)
  - Supports SDRAM, Static Memory, NAND Flash and CompactFlash®
- LCD Controller
  - Supports Passive or Active Displays
  - Up to 16-bits per Pixel in STN Color Mode
  - Up to 16M Colors in TFT Mode (24-bit per Pixel), Resolution up to 2048 x 2048
- USB
  - USB 2.0 Full Speed (12 Mbits per second) Host Double Port
    Dual On-chip Transceivers
    Integrated FIFOs and Dedicated DMA Channels
  - USB 2.0 Full Speed (12 Mbits per second) Device Port
    On-chip Transceiver, 2-Kbyte Configurable Integrated FIFOs
- Bus Matrix
  - Handles Five Masters and Five Slaves
  - Boot Mode Select Option
  - Remap Command
- Fully Featured System Controller (SYSC) for Efficient System Management, including
  - Reset Controller, Shutdown Controller, Four 32-bit Battery Backup Registers for a Total of 16 Bytes
  - Clock Generator and Power Management Controller
  - Advanced Interrupt Controller and Debug Unit
  - Periodic Interval Timer, Watchdog Timer and Real-time Timer
  - Three 32-bit PIO Controllers
- Reset Controller (RSTC)
  - Based on Power-on Reset Cells, Reset Source Identification and Reset Output Control
- Shutdown Controller (SHDWC)
  - Programmable Shutdown Pin Control and Wake-up Circuitry
- Clock Generator (CKGR)
  - 32.768 kHz Low-power Oscillator on Battery Backup Power Supply, Providing a Permanent Slow Clock
  - 3 to 20 MHz On-chip Oscillator and two PLLs
- Power Management Controller (PMC)
  - Very Slow Clock Operating Mode, Software Programmable Power Optimization Capabilities
  - Four Programmable External Clock Signals


6062B–ATARM–15-Nov-05



- Advanced Interrupt Controller (AIC)
  - Individually Maskable, Eight-level Priority, Vectored Interrupt Sources
  - Three External Interrupt Sources and One Fast Interrupt Source, Spurious Interrupt Protected
- Debug Unit (DBGU)
  - 2-wire USART and support for Debug Communication Channel, Programmable ICE Access Prevention
- Periodic Interval Timer (PIT)
  - 20-bit Interval Timer plus 12-bit Interval Counter
- Watchdog Timer (WDT)
  - Key Protected, Programmable Only Once, Windowed 12-bit Counter, Running at Slow Clock
- Real-Time Timer (RTT)
  - 32-bit Free-running Backup Counter Running at Slow Clock
- Three 32-bit Parallel Input/Output Controllers (PIO) PIOA, PIOB and PIOC
  - 96 Programmable I/O Lines Multiplexed with up to Two Peripheral I/Os
  - Input Change Interrupt Capability on Each I/O Line
  - Individually Programmable Open-drain, Pull-up Resistor and Synchronous Output
- Nineteen Peripheral DMA (PDC) Channels
- Multimedia Card Interface (MCI)
  - Compliant with Multimedia Cards and SDCards
  - Automatic Protocol Control and Fast Automatic Data Transfers with PDC, MMC and SDCard Compliant
- Three Synchronous Serial Controllers (SSC)
  - Independent Clock and Frame Sync Signals for Each Receiver and Transmitter
  - I²S Analog Interface Support, Time Division Multiplex Support
  - High-speed Continuous Data Stream Capabilities with 32-bit Data Transfer
- Three Universal Synchronous/Asynchronous Receiver Transmitters (USART)
  - Individual Baud Rate Generator, IrDA Infrared Modulation/Demodulation
  - Support for ISO7816 T0/T1 Smart Card, Hardware and Software Handshaking, RS485 Support
- Two Master/Slave Serial Peripheral Interface (SPI)
  - 8- to 16-bit Programmable Data Length, Four External Peripheral Chip Selects
- One Three-channel 16-bit Timer/Counters (TC)
  - Three External Clock Inputs, Two multi-purpose I/O Pins per Channel
  - Double PWM Generation, Capture/Waveform Mode, Up/Down Capability
- Two-wire Interface (TWI)
  - Master Mode Support, All Two-wire Atmel EEPROMs Supported
- IEEE 1149.1 JTAG Boundary Scan on All Digital Pins
- Required Power Supplies:
  - 1.08V to 1.32V for VDDCORE and VDDBU
  - 2.7V to 3.6V for VDDOSC and for VDDPLL
  - 2.7V to 3.6V for VDDIOP (Peripheral I/Os) and for VDDIOM (Memory I/Os)
- Available in a 217-ball LFBGA RoHS-compliant Package

1. Description

The AT91SAM9261 is a complete system-on-chip built around the ARM926EJ-S ARM Thumb processor with an extended DSP instruction set and Jazelle Java accelerator. It achieves 200 MIPS at 180 MHz.

The AT91SAM9261 is an optimized host processor for applications with an LCD display. Its integrated LCD controller supports BW and up to 16M color, active and passive LCD displays. The 160-kbyte integrated SRAM can be configured as a frame buffer minimizing the impact for LCD refresh on the overall processor performance. The External Bus Interface incorporates controllers for synchronous DRAM (SDRAM) and Static memories and features specific interface circuitry for CompactFlash and NAND Flash.

The AT91SAM9261 integrates a ROM-based Boot Loader supporting code shadowing from, for example, external DataFlash® into external SDRAM. The software controlled Power Management Controller (PMC) keeps system power consumption to a minimum by selectively enabling/disabling the processor and various peripherals and adjustment of the operating frequency.

The AT91SAM9261 also benefits from the integration of a wide range of debug features including JTAG-ICE, a dedicated UART debug channel (DBGU) and an embedded real time trace. This enables the development and debug of all applications, especially those with real-time constraints.

2. Block Diagram

Figure 2-1. AT91SAM9261 Block Diagram

3. Signal Description

Table 3-1. Signal Description by Peripheral

Signal Name	Function	Type	Active Level	Comments
Power				
VDDIOM	EBI I/O Lines Power Supply	Power		2.7V to 3.6V
VDDIOP	Peripherals I/O Lines Power Supply	Power		2.7V to 3.6V
VDDBU	Backup I/O Lines Power Supply	Power		1.08V to 1.32V
VDDPLL	PLL Power Supply	Power		2.7V to 3.6V
VDDOSC	Oscillator Power Supply	Power		2.7V to 3.6V
VDDCORE	Core Chip Power Supply	Power		1.08V to 1.32V
GND	Ground	Ground		
GNDPLL	PLL Ground	Ground		
GNDOSC	Oscillator Ground	Ground		
GNDBU	Backup Ground	Ground		
Clocks, Oscillators and PLLs				
XIN	Main Oscillator Input	Input		
XOUT	Main Oscillator Output	Output		
XIN32	Slow Clock Oscillator Input	Input		
XOUT32	Slow Clock Oscillator Output	Output		
PLLRCA	PLL Filter	Input		
PLLRCB	PLL Filter	Input		
PCK0 - PCK3	Programmable Clock Output	Output		
Shutdown, Wakeup Logic				
SHDN	Shutdown Control	Output		Driven at 0V only. Do not tie over VDDBU
WKUP	Wake-Up Input	Input		Accept between 0V and VDDBU.
ICE and JTAG				
TCK	Test Clock	Input		No pull-up resistor.
RTCK	Returned Test Clock	Output		No pull-up resistor.
TDI	Test Data In	Input		No pull-up resistor.
TDO	Test Data Out	Output		
TMS	Test Mode Select	Input		No pull-up resistor.
NTRST	Test Reset Signal	Input	Low	Pull-up resistor.
JTAGSEL	JTAG Selection	Input		Pull-down resistor.
ETM				
TSYNC	Trace Synchronization Signal	Output		
TCLK	Trace Clock	Output		
TPS0 - TPS2	Trace ARM Pipeline Status	Output		

Table 3-1. Signal Description by Peripheral (Continued)

Signal Name	Function	Type	Active Level	Comments
TPK0 - TPK15	Trace Packet Port	Output		
Reset/Test				
NRST	Microcontroller Reset	I/O	Low	Pull-up resistor
TST	Test Mode Select	Input		Pull-down resistor.
BMS	Boot Mode Select	Input		
Debug Unit				
DRXD	Debug Receive Data	Input		
DTXD	Debug Transmit Data	Output		
AIC				
IRQ0 - IRQ2	External Interrupt Inputs	Input		
FIQ	Fast Interrupt Input	Input		
PIO				
PA0 - PA31	Parallel IO Controller A	I/O		Pulled-up input at reset
PB0 - PB31	Parallel IO Controller B	I/O		Pulled-up input at reset
PC0 - PC31	Parallel IO Controller C	I/O		Pulled-up input at reset
EBI				
D0 - D31	Data Bus	I/O		Pulled-up input at reset
A0 - A25	Address Bus	Output		0 at reset
NWAIT	External Wait Signal	Input	Low	
SMC				
NCS0 - NCS7	Chip Select Lines	Output	Low	
NWR0 - NWR3	Write Signal	Output	Low	
NRD	Read Signal	Output	Low	
NWE	Write Enable	Output	Low	
NBS0 - NBS3	Byte Mask Signal	Output	Low	
CompactFlash Support				
CFCE1 - CFCE2	CompactFlash Chip Enable	Output	Low	
CFOE	CompactFlash Output Enable	Output	Low	
CFWE	CompactFlash Write Enable	Output	Low	
CFIOR	CompactFlash IO Read	Output	Low	
CFIOW	CompactFlash IO Write	Output	Low	
CFRNW	CompactFlash Read Not Write	Output		
CFCS0 - CFCS1	CompactFlash Chip Select Lines	Output	Low	
NAND Flash Support				
NANDOE	NAND Flash Output Enable	Output	Low	
NANDWE	NAND Flash Write Enable	Output	Low	

Table 3-1. Signal Description by Peripheral (Continued)

Signal Name	Function	Type	Active Level	Comments
NANDCS	NAND Flash Chip Select	Output	Low	
SDRAM Controller				
SDCK	SDRAM Clock	Output		
SDCKE	SDRAM Clock Enable	Output	High	
SDCS	SDRAM Controller Chip Select	Output	Low	
BA0 - BA1	Bank Select	Output		
SDWE	SDRAM Write Enable	Output	Low	
RAS - CAS	Row and Column Signal	Output	Low	
SDA10	SDRAM Address 10 Line	Output		
Multimedia Card Interface				
MCCK	Multimedia Card Clock	Output		
MCCDA	Multimedia Card A Command	I/O		
MCDA0 - MCDA3	Multimedia Card A Data	I/O		
USART				
SCK0 - SCK2	Serial Clock	I/O		
TXD0 - TXD2	Transmit Data	Output		
RXD0 - RXD2	Receive Data	Input		
RTS0 - RTS2	Request To Send	Output		
CTS0 - CTS2	Clear To Send	Input		
Synchronous Serial Controller				
TD0 - TD2	Transmit Data	Output		
RD0 - RD2	Receive Data	Input		
TK0 - TK2	Transmit Clock	I/O		
RK0 - RK2	Receive Clock	I/O		
TF0 - TF2	Transmit Frame Sync	I/O		
RF0 - RF2	Receive Frame Sync	I/O		
Timer/Counter				
TCLK0 - TCLK2	External Clock Input	Input		
TIOA0 - TIOA2	I/O Line A	I/O		
TIOB0 - TIOB2	I/O Line B	I/O		
SPI				
SPI0_MISO - SPI1_MISO	Master In Slave Out	I/O		
SPI0_MOSI - SPI1_MOSI	Master Out Slave In	I/O		
SPI0_SPCK - SPI1_SPCK	SPI Serial Clock	I/O		



Table 3-1. Signal Description by Peripheral (Continued)

Signal Name	Function	Type	Active Level	Comments
SPI0_NPCS0, SPI1_NPCS0	SPI Peripheral Chip Select 0	I/O	Low	
SPI0_NPCS1 - SPI0_NPCS3 SPI1_NPCS1 - SPI1_NPCS3	SPI Peripheral Chip Select	Output	Low	
Two-Wire Interface				
TWD	Two-wire Serial Data	I/O		
TWCK	Two-wire Serial Clock	I/O		
LCD Controller				
LCDD0 - LCDD23	LCD Data Bus	Output		
LCDVSYNC	LCD Vertical Synchronization	Output		
LCDHSYNC	LCD Horizontal Synchronization	Output		
LCDDOTCK	LCD Dot Clock	Output		
LCDDEN	LCD Data Enable	Output		
LCDCC	LCD Contrast Control	Output		
USB Device Port				
DDM	USB Device Port Data -	Analog		
DDP	USB Device Port Data +	Analog		
USB Host Port				
HDMA	USB Host Port A Data -	Analog		
HDPA	USB Host Port A Data +	Analog		
HDMB	USB Host Port B Data -	Analog		
HDPB	USB Host Port B Data +	Analog		

4. Package and Pinout

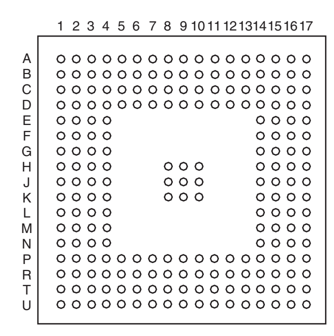
The ATSAM9261 is available in a 217-ball LFBGA RoHS-compliant package, 15 x 15 mm, 0.8 mm ball pitch

4.1 Mechanical Overview of the 217-ball LFBGA Package

Figure 4-1 shows the orientation of the 217-ball LFBGA Package.

A detailed mechanical description is given in the Mechanical Characteristics section of the product datasheet.

Figure 4-1. 217-ball LFBGA Pinout (Top View)



4.2 217-ball LFBGA Package Pinout

Table 4-1. AT91SAM9261 Pinout for 217-ball LFBGA Package [(1)]

Pin	Signal Name
A1	A19
A2	A16/BA0
A3	A14
A4	A12
A5	A9
A6	A6
A7	A3
A8	A2
A9	NC
A10	XOUT32
A11	XIN32
A12	DDP
A13	HDPB
A14	HDMB
A15	PB27
A16	GND
A17	PB24
B1	A20
B2	A18
B3	A15
B4	A13
D5	VDDCORE
D6	A10
D7	A5
D8	A0/NBS0
D9	SHDW
D10	NC
D11	VDDIOP
D12	PB29
D13	PB28
D14	PB23
D15	PB20
D16	PB17
D17	TCK
E1	NWR1/NBS1/CFIOR
E2	NWR0/NWE/CFWE
E3	NRD/CFOE
E4	SDA10
E14	PB22
E15	PB18
E16	PB15
E17	TDI
J14	VDDIOP
J15	PB9
J16	PB6
J17	PB4
K1	D6
K2	D8
K3	D10
K4	D7
K8	GND
K9	GND
K10	GND
K14	VDDCORE
K15	PB3/BMS
K16	PB1
K17	PB2
L1	D9
L2	D11
L3	D12
L4	VDDIOM
L14	PA30
L15	PA27
P17	PA20
R1	PC19
R2	PC21
R3	GND
R4	PC27
R5	PC29
R6	PC4
R7	PC8
R8	PC12
R9	PC14
R10	VDDPLL
R11	PA0
R12	PA7
R13	PA10
R14	PA13
R15	PA17
R16	GND
R17	PA18
T1	PC20
T2	PC23
T3	PC26



Table 4-1. AT91SAM9261 Pinout for 217-ball LFBGA Package [(1)] (Continued)

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
B5	A11	F1	SDCKE	L16	PA31	T4	PC2
B6	A7	F2	RAS	L17	PB0	T5	VDDIOP
B7	A4	F3	NWR3/NBS3/CFIOW	M1	D13	T6	PC5
B8	A1/NBS2/NWR2	F4	NCS0	M2	D15	T7	PC9
B9	VDDBU	F14	PB16	M3	PC18	T8	PC10
B10	JTAGSEL	F15	NRST	M4	VDDCORE	T9	PC15
B11	WKUP	F16	TDO	M14	PA25	T10	VDDOSC
B12	DDM	F17	NTRST	M15	PA26	T11	GNDOSC
B13	PB31	G1	D0	M16	PA28	T12	PA1
B14	HDMA	G2	D1	M17	PA29	T13	PA4
B15	PB26	G3	SDWE	N1	D14	T14	PA6
B16	PB25	G4	NCS3/NANDCS	N2	PC17	T15	PA8
B17	PB19	G14	PB14	N3	PC31	T16	PA11
C1	A22	G15	PB12	N4	VDDIOM	T17	PA14
C2	A21	G16	PB11	N14	PA22	U1	PC25
C3	VDDIOM	G17	PB8	N15	PA21	U2	PC0
C4	A17/BA1	H1	D2	N16	PA23	U3	PC3
C5	VDDIOM	H2	D3	N17	PA24	U4	GND
C6	A8	H3	VDDIOM	P1	PC16	U5	PC6
C7	GND	H4	SDCK	P2	PC30	U6	VDDIOP
C8	VDDIOM	H8	GND	P3	PC22	U7	GND
C9	GNDBU	H9	GND	P4	PC24	U8	PC13
C10	TST	H10	GND	P5	PC28	U9	PLLRCB
C11	GND	H14	PB10	P6	PC1	U10	PLLRCA
C12	HDPA	H15	PB13	P7	PC7	U11	XIN
C13	PB30	H16	PB7	P8	PC11	U12	XOUT
C14	NC	H17	PB5	P9	GNDPLL	U13	PA2
C15	VDDIOP	J1	D4	P10	PA3	U14	PA5
C16	PB21	J2	D5	P11	VDDIOP	U15	PA12
C17	TMS	J3	GND	P12	VDDCORE	U16	PA9
D1	NCS2	J4	CAS	P13	PA15	U17	RTCK
D2	NCS1/SDCS	J8	GND	P14	PA16		
D3	GND	J9	GND	P15	VDDIOP		
D4	VDDIOM	J10	GND	P16	PA19		

Note: 1. Shaded cells define the pins powered by VDDIOM.

5. Power Considerations

5.1 Power Supplies

The AT91SAM9261 has six types of power supply pins:

- VDDCORE pins: Power the core, including the processor, the memories and the peripherals; voltage ranges from 1.08V and 1.32V, 1.2V nominal.
- VDDIOM pins: Power the External Bus Interface I/O lines; voltage ranges from 2.7V and 3.6V, 3.3V nominal.
- VDDIOP pins: Power the Peripheral I/O lines and the USB transceivers; voltage ranges from 2.7V and 3.6V, 3.3V nominal.
- VDDBU pin: Powers the Slow Clock oscillator and a part of the System Controller; voltage ranges from 1.08V and 1.32V, 1.2V nominal.
- VDDPLL pin: Powers the PLL cells; voltage ranges from 2.7V and 3.6V, 3V or 3.3V nominal.
- VDDOSC pin: Powers the Main Oscillator cells; voltage ranges from 2.7V and 3.6V, 3V or 3.3V nominal.

The double power supplies VDDIOM and VDDIOP are identified in Table 4-1 on page 9. These supplies enable the user to power the device differently for interfacing with memories and for interfacing with peripherals.

Ground pins GND are common to VDDCORE, VDDIOM and VDDIOP pins power supplies. Separated ground pins are provided for VDDBU, VDDOSC and VDDPLL. The ground pins are respectively GNDBU, GNDOSC and GNDPLL.

5.2 Power Consumption

The AT91SAM9261 consumes about 550 µA of static current on VDDCORE at 25°C. This static current rises at up to 5.5 mA if the temperature increases to 85°C.

On VDDBU, the current does not exceed 3 µA @25°C, but can rise at up to 20 µA @85°C.

For dynamic power consumption, the AT91SAM9261 consumes a maximum of 50 mA on VDDCORE at maximum speed in typical conditions (1.2V, 25°C), processor running full-performance algorithm.

6. I/O Line Considerations

6.1 JTAG Port Pins

TMS, TDI and TCK are schmitt trigger inputs and have no pull-up resistors.

TDO and RTCK are outputs, driven at up to VDDIOP, and have no pull-up resistor.

The JTAGSEL pin is used to select the JTAG boundary scan when asserted at a high level. It integrates a permanent pull-down resistor of about 15 kΩ to GNDBU, so that it can be left unconnected for normal operations.

The NTRST pin is used to initialize the embedded ICE TAP Controller when asserted at a low level. It integrates a permanent pull-up resistor of about 15 kΩ to VDDIOP, so that it can be left unconnected for normal operations.

6.2 Test Pin

The TST pin is used for manufacturing test purposes when asserted high. It integrates a permanent pull-down resistor of about 15 kΩ to GNDBU, so that it can be left unconnected for normal operations. Driving this line at a high level leads to unpredictable results.

6.3 Reset Pin

NRST is an open-drain output integrating a non-programmable pull-up resistor. It can be driven with voltage at up to VDDIOP. As the product integrates power-on reset cells, the NRST pin can be left unconnected in case no reset from the system needs to be applied to the product.

The NRST pin integrates a permanent pull-up resistor of 100 kΩ minimum to VDDIOP.

The NRST signal is inserted in the Boundary Scan.

6.4 PIO Controller A, B and C Lines

All the I/O lines PA0 to PA31, PB0 to PB31, and PC0 to PC31 integrate a programmable pull-up resistor of 100 kΩ. Programming of this pull-up resistor is performed independently for each I/O line through the PIO Controllers.

After reset, all the I/O lines default as inputs with pull-up resistors enabled, except those which are multiplexed with the External Bus Interface signals that require to be enabled as Peripherals at reset. This is explicitly indicated in the column “Reset State” of the PIO Controller multiplexing tables.

6.5 Shutdown Logic pins

The SHDN pin is an output only, which is driven by the product only at low level. It can be tied high with an external pull-up resistor at VDDBU only.

The pin WKUP is an input-only. It can accept voltages only between 0V and VDDBU.

7. Processor and Architecture

7.1 ARM926EJ-S Processor

- RISC Processor Based on ARM v5TEJ Architecture with Jazelle technology for Java acceleration
- Two Instruction Sets
  - ARM High-performance 32-bit Instruction Set
  - Thumb High Code Density 16-bit Instruction Set
- DSP Instruction Extensions
- 5-Stage Pipeline Architecture:
  - Instruction Fetch (F)
  - Instruction Decode (D)
  - Execute (E)
  - Data Memory (M)
  - Register Write (W)
- 16-Kbyte Data Cache, 16-Kbyte Instruction Cache
  - Virtually-addressed 4-way Associative Cache
  - Eight words per line
  - Write-through and Write-back Operation
  - Pseudo-random or Round-robin Replacement
- Write Buffer
  - Main Write Buffer with 16-word Data Buffer and 4-address Buffer
  - DCache Write-back Buffer with 8-word Entries and a Single Address Entry
  - Software Control Drain
- Standard ARM v4 and v5 Memory Management Unit (MMU)
  - Access Permission for Sections
  - Access Permission for large pages and small pages can be specified separately for each quarter of the page
  - 16 embedded domains
- Bus Interface Unit (BIU)
  - Arbitrates and Schedules AHB Requests
  - Separate Masters for both instruction and data access providing complete AHB system flexibility
  - Separate Address and Data Buses for both the 32-bit instruction interface and the 32-bit data interface
  - On Address and Data Buses, data can be 8-bit (Bytes), 16-bit (Half-words) or 32-bit (Words)

7.2 Debug and Test Features

- Integrated Embedded In-circuit Emulator Real-Time
  - Two real-time Watchpoint Units
  - Two Independent Registers: Debug Control Register and Debug Status Register
  - Test Access Port Accessible through JTAG Protocol
  - Debug Communications Channel
- Debug Unit
  - Two-pin UART
  - Debug Communication Channel Interrupt Handling
  - Chip ID Register
- Embedded Trace Macrocell: ETM9™
  - Medium+ Level Implementation
  - Half-rate Clock Mode
  - Four Pairs of Address Comparators
  - Two Data Comparators
  - Eight Memory Map Decoder Inputs
  - Two 16-bit Counters
  - One 3-stage Sequencer
  - One 45-byte FIFO
- IEEE1149.1 JTAG Boundary-scan on All Digital Pins

7.3 Bus Matrix

- Five Masters and Five Slaves handled
  - Handles Requests from the ARM926EJ-S, USB Host Port, LCD Controller and the Peripheral DMA Controller to internal ROM, internal SRAM, EBI, APB, LCD Controller and USB Host Port.
  - Round-Robin Arbitration (three modes supported: no default master, last accessed default master, fixed default master)
  - Burst Breaking with Slot Cycle Limit
- One Address Decoder Provided per Master
  - Three different slaves may be assigned to each decoded memory area: one for internal boot, one for external boot, one after remap.
- Boot Mode Select Option
  - Non-volatile Boot Memory can be Internal or External.
  - Selection is made by BMS pin sampled at reset.
- Remap Command
  - Allows Remapping of an Internal SRAM in Place of the Boot Non-Volatile Memory
  - Allows Handling of Dynamic Exception Vectors

7.4 Peripheral DMA Controller

- Transfers from/to peripheral to/from any memory space without intervention of the processor.
- Next Pointer Support, forbids strong real-time constraints on buffer management.
- Nineteen channels
  - Two for each USART
  - Two for the Debug Unit
  - Two for each Serial Synchronous Controller
  - Two for each Serial Peripheral Interface
  - One for the Multimedia Card Interface

8. Memories

8.1 Embedded Memories

- 32K-byte ROM
  - Single Cycle Access at full bus speed
- 160K-byte Fast SRAM
  - Single Cycle Access at full bus speed
  - Supports ARM926EJ-S TCM interface at full processor speed

8.2 Memory Mapping

A first level of address decoding is performed by the Bus Matrix, i.e., the implementation of the Advanced High performance Bus (AHB) for its Master and Slave interfaces with additional features.

Decoding breaks up the 4G bytes of address space into 16 areas of 256M bytes. The areas 1 to 8 are directed to the EBI that associates these areas to the external chip selects NCS0 to NCS7. The area 0 is reserved for the addressing of the internal memories, and a second level of decoding provides 1M byte of internal memory area. The area 15 is reserved for the peripherals and provides access to the Advanced Peripheral Bus (APB).

Other areas are unused and performing an access within them provides an abort to the master requesting such an access.

Figure 8-1. Product Memory Mapping

Size	Address range	Area
256M Bytes	**0x0000 0000** 0x0FFF FFFF	Internal Memories
8 x 256M Bytes 2,048M bytes	**0x1000 0000** 0x8FFF FFFF	External Bus Interface Chip Select 0 to 7
6 x 256M Bytes 1,536M Bytes	**0x9000 0000** 0xEFFF FFFF	Undefined (Abort)
256M Bytes	**0xF000 0000** 0xFFFF FFFF	Internal Peripherals

The Bus Matrix manages five Masters and five Slaves.

Each Master has its own bus and its own decoder, thus allowing a different memory mapping per Master.

Regarding Master 0 and Master 1 (ARM926 Instruction and Data), three different Slaves are assigned to the memory space decoded at address 0x0: one for internal boot, one for external boot, one after remap, refer to Table 8-3 for details.

Table 8-1. List of Bus Matrix Masters

Master 0	ARM926 Instruction
Master 1	ARM926 Data
Master 2	PDC
Master 3	LCD Controller
Master 4	USB Host

Each Slave has its own arbiter, thus allowing a different arbitration per Slave.

Table 8-2. List of Bus Matrix Slaves

Slave 0	Internal SRAM
Slave 1	Internal ROM
Slave 2	LCD Controller and USB Host Port Interfaces
Slave 3	External Bus Interface
Slave 4	Internal Peripherals

8.2.1 External Memory Mapping

The external memories are accessed through the External Bus Interface (Bus Matrix Slave 3).

Figure 8-2. External Memory Mapping

256M Bytes	0x1000 0000	0x1FFF FFFF	Chip Select 0	SMC	
256M Bytes	0x2000 0000	0x2FFF FFFF	Chip Select 1	SMC or SDRAMC	
256M Bytes	0x3000 0000	0x3FFF FFFF	Chip Select 2	SMC	
256M Bytes	0x4000 0000	0x4FFF FFFF	Chip Select 3	SMC	NAND Flash
256M Bytes	0x5000 0000	0x5FFF FFFF	Chip Select 4	SMC	CompactFlash Slot 0
256M Bytes	0x6000 0000	0x6FFF FFFF	Chip Select 5	SMC	CompactFlash Slot 1
256M Bytes	0x7000 0000	0x7FFF FFFF	Chip Select 6	SMC	
256M Bytes	0x8000 0000	0x8FFF FFFF	Chip Select 7	SMC	

8.2.2 Internal Memory Mapping

Table 8-3 summarizes the Internal Memory Mapping for each Master, depending on the Remap status and the BMS state at reset.

Table 8-3. Internal Memory Mapping

Address	Master 0: ARM926 Instruction			Master 1: ARM926 Data			Master 2: PDC	Master 3: LCD Controller	Master 4: USB Host
	REMAP (RCB0) = 0		REMAP (RCB0) = 1	REMAP (RCB1) = 0		REMAP (RCB1) = 1			
	BMS = 1	BMS = 0		BMS = 1	BMS = 0				
0x0000 0000	Int. ROM	EBI NCS0	Int. RAM C	Int. ROM	EBI NCS0	Int. RAM C			
0x0010 0000	Int. RAM A	Int. RAM A	Int. RAM A	Int. RAM A	Int. RAM A	Int. RAM A	Int. RAM A	Int. RAM A	Int. RAM A
0x0020 0000	Int. RAM B	Int. RAM B	Int. RAM B	Int. RAM B	Int. RAM B	Int. RAM B	Int. RAM B	Int. RAM B	Int. RAM B
0x0030 0000	Int. RAM C	Int. RAM C	Int. RAM C	Int. RAM C	Int. RAM C	Int. RAM C	Int. RAM C	Int. RAM C	Int. RAM C
0x0040 0000	Int. ROM	Int. ROM	Int. ROM	Int. ROM	Int. ROM	Int. ROM	Int. ROM	Int. ROM	Int. ROM
0x0050 0000	-	-	-	UHP I/F	UHP I/F	UHP I/F	-	-	-
0x0060 0000	-	-	-	LCD I/F	LCD I/F	LCD I/F	-	-	-

8.2.2.1 Internal SRAM

The AT91SAM9261 embeds a high-speed 160-Kbyte SRAM. This Internal SRAM is split into three areas. Its Memory Mapping is detailed in Table 8-3 above.

- Internal SRAM A is the ARM926EJ-S Instruction TCM and the user can map this SRAM block anywhere in the ARM926 instruction memory space using CP15 instructions. This SRAM block is also accessible by the ARM926 Data Master and by the AHB Masters through the AHB bus at address 0x0010 0000.
- Internal SRAM B is the ARM926EJ-S Data TCM and the user can map this SRAM block anywhere in the ARM926 data memory space using CP15 instructions. This SRAM block is also accessible by the ARM926 Data Master and by the AHB Masters through the AHB bus at address 0x0020 0000.
- Internal SRAM C is only accessible by all the AHB Masters.
  After reset and until the Remap Command is performed, this SRAM block is accessible through the AHB bus at address 0x0030 0000 by all the AHB Masters. After Remap, this SRAM block also becomes accessible through the AHB bus at address 0x0 by the ARM926 Instruction and the ARM926 Data Masters.

Within the 160-Kbyte SRAM size available, the amount of memory assigned to each block is software programmable as a multiple of 16K Bytes according to Table 8-4. This

table provides the size of the Internal SRAM C according to the size of the Internal SRAM A and the Internal SRAM B.

Table 8-4. Internal SRAM Block Size

Internal SRAM C		Internal SRAM A (ITCM) Size			
		0	16K Bytes	32K Bytes	64K Bytes
Internal SRAM B (DTCM) size	0	160K Bytes	144K Bytes	128K Bytes	96K Bytes
	16K Bytes	144K Bytes	128K Bytes	112K Bytes	80K Bytes
	32K Bytes	128K Bytes	112K Bytes	96K Bytes	64K Bytes
	64K Bytes	96K Bytes	80K Bytes	64K Bytes	32K Bytes

Note that among the ten 16-Kbyte blocks making up the Internal SRAM, two are permanently assigned to Internal SRAM C.

At reset, the whole memory (160K Bytes) is assigned to Internal SRAM C.

The memory blocks assigned to SRAM A, SRAM B and SRAM C areas are not contiguous and when the user dynamically changes the Internal SRAM configuration, the new 16-Kbyte block organization may affect the previous configuration from a software point of view.

Table 8-5 illustrates different configurations and the related 16-Kbyte blocks (RB0 to RB9) assignments.

Table 8-5. 16-Kbyte Block Allocation

Decoded Area	Address	Configuration examples and related 16-Kbyte block assignments			
		ITCM = 0 Kbyte DTCM = 0 Kbyte AHB = 160K Bytes (1)	ITCM = 64K Bytes DTCM = 64K Bytes AHB = 32K Bytes	ITCM = 32K Bytes DTCM = 64K Bytes AHB = 64K Bytes	ITCM = 32K Bytes DTCM = 16K Bytes AHB = 112K Bytes
Internal SRAM A (ITCM)	0x0010 0000		RB3	RB3	RB3
	0x0010 4000		RB2	RB2	RB2
	0x0010 8000		RB1		
	0x0010 C000		RB0		
Internal SRAM B (DTCM)	0x0020 0000		RB7	RB7	RB7
	0x0020 4000		RB6	RB6	
	0x0020 8000		RB5	RB5	
	0x0020 C000		RB4	RB4	

Table 8-5. 16-Kbyte Block Allocation (Continued)

Decoded Area	Address	Configuration examples and related 16-Kbyte block assignments			
		ITCM = 0 Kbyte DTCM = 0 Kbyte AHB = 160K Bytes [(1)]	**ITCM = 64K Bytes DTCM = 64K Bytes AHB = 32K Bytes**	**ITCM = 32K Bytes DTCM = 64K Bytes AHB = 64K Bytes**	**ITCM = 32K Bytes DTCM = 16K Bytes AHB = 112K Bytes**
Internal SRAM C (AHB)	0x0030 0000	RB9	RB9	RB9	RB9
	0x0030 4000	RB8	RB8	RB8	RB8
	0x0030 8000	RB7		RB1	RB6
	0x0030 C000	RB6		RB0	RB5
	0x0031 0000	RB5			RB4
	0x0031 4000	RB4			RB1
	0x0031 8000	RB3			RB0
	0x0031 C000	RB2			
	0x0032 0000	RB1			
	0x0032 4000	RB0			

Note: 1. Configuration after reset.

8.2.2.2 *Internal ROM*

The AT91SAM9261 integrates 32-Kbyte Internal ROM mapped at address 0x0040 0000. It is also accessible at address 0x0 after reset and before Remap if the BMS is tied high during reset.

8.2.2.3 *USB Host Port*

The AT91SAM9261 integrates a USB Host Port Open Host Controller Interface (OHCI). The registers of this interface are directly accessible on the AHB Bus and are mapped like a standard internal memory at address 0x0050 0000.

8.2.2.4 *LCD Controller*

The AT91SAM9261 integrates an LCD Controller. The interface is directly accessible on the AHB Bus and is mapped like a standard internal memory at address 0x0060 0000.

8.2.3 Boot Program

- DataFlash Boot
  - Downloads and runs an application from SPI DataFlash into internal SRAM
  - Downloaded code size from SPI DataFlash depends on embedded SRAM size
  - Automatic detection of valid application
  - SPI DataFlash connected to SPI NPCS0
- Boot Uploader in case no valid program is detected in external SPI DataFlash
  - Small monitor functionalities (read/write/run) interface with SAMBA application
  - Automatic detection of the communication link

    Serial communication on a DBGU (XModem protocol)

    USB Device Port (CDC Protocol)



8.2.4 ETM Memories

The eight ETM9 Medium+ memory map decoder inputs are connected to custom address decoders and the resulting memory mapping is summarized in Table 8-6.

Table 8-6. ETM9 Memory Mapping

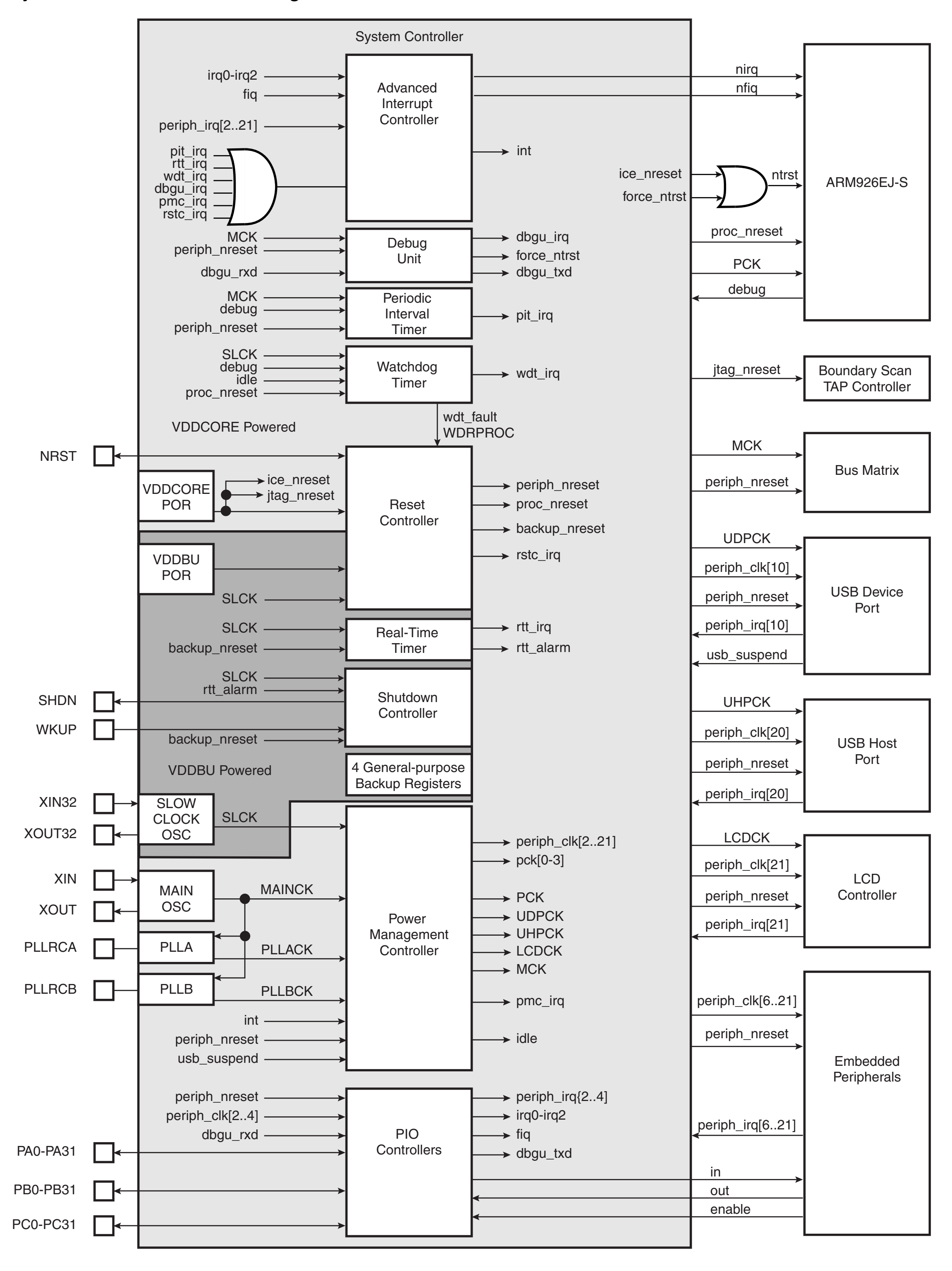
Product Resource	Area	Access Type	Start Address	End Address
SRAM	Internal	Data	0x0000 0000	0x002F FFFF
SRAM	Internal	Fetch	0x0000 0000	0x002F FFFF
ROM	Internal	Data	0x0040 0000	0x004F FFFF
ROM	Internal	Fetch	0x0040 0000	0x004F FFFF
External Bus Interface	External	Data	0x1000 0000	0x8FFF FFFF
External Bus Interface	External	Fetch	0x1000 0000	0x8FFF FFFF
User Peripherals	Internal	Data	0xF000 0000	0xFFFF BFFF
System Peripherals	Internal	Data	0xFFFF C000	0xFFFF FFFF

9. System Controller

The System Controller manages all vital blocks of the microcontroller: interrupts, clocks, power, time, debug and reset.

9.1 Block Diagram

Figure 9-1. System Controller Block Diagram



9.1.1 System Controller Mapping

The System Peripherals are all mapped within the highest 6K bytes of address space, between addresses 0xFFFF EA00 and 0xFFFF FFFF. Each peripheral has an address space of 256 or 512 Bytes, representing 64 or 128 registers.

Figure 9-2 shows the mapping of the EBI Peripherals, the Matrix and the System Controller.

Figure 9-2. System Controller Mapping

Address		Peripheral Name	Size
0xFFFF EA00 – 0xFFFF EBFF	SDRAMC	SDRAM Controller	512 bytes/128 words
0xFFFF EC00 – 0xFFFF EDFF	SMC	Static Memory Controller	512 bytes/128 words
0xFFFF EE00 – 0xFFFF EFFF	MATRIX	Bus Matrix	512 bytes/128 words
0xFFFF F000 – 0xFFFF F1FF	AIC	Advanced Interrupt Controller	512 bytes/128 words
0xFFFF F200 – 0xFFFF F3FF	DBGU	Debug Unit	512 bytes/128 words
0xFFFF F400 – 0xFFFF F5FF	PIOA	Parallel I/O Controller A	512 bytes/128 words
0xFFFF F600 – 0xFFFF F7FF	PIOB	Parallel I/O Controller B	512 bytes/128 words
0xFFFF F800 – 0xFFFF F9FF	PIOC	Parallel I/O Controller C	512 bytes/128 words
0xFFFF FA00 – 0xFFFF FBFF	Reserved		512 bytes/128 words
0xFFFF FC00 – 0xFFFF FCFF	PMC	Power Management Controller	256 bytes/64 words
0xFFFF FD00 – 0xFFFF FD0F	RSTC	Reset Controller	16 bytes/4 words
0xFFFF FD10 – 0xFFFF FD1F	SHDC	Shutdown Controller	16 bytes/4 words
0xFFFF FD20 – 0xFFFF FD2F	RTTC	Real-Time Timer Controller	16 bytes/4 words
0xFFFF FD30 – 0xFFFF FD3F	PITC	Periodic Interval Timer Controller	16 bytes/4 words
0xFFFF FD40 – 0xFFFF FD4F	WDTC	Watchdog Timer Controller	16 bytes/4 words
0xFFFF FD50 – 0xFFFF FD5F	GPBR	General-purpose Backup Registers	16 bytes/4 words
0xFFFF FD60 – 0xFFFF FDFF	Reserved		
0xFFFF FE00 – 0xFFFF FEFF	Reserved		256 bytes/64 words
0xFFFF FF00 – 0xFFFF FFFF	Reserved		256 bytes/64 words

9.2 Reset Controller

- Based on two Power-on-Reset cells
- Status of the last reset
  - Either cold reset, first reset, soft reset, user reset, watchdog reset, wake-up reset
- Controls the internal resets and the NRST pin output.

9.3 Shutdown Controller

- Shutdown and Wake-up logic:
  - Software programmable assertion of the SHDN open-drain pin.
  - Deassertion Programmable on a WKUP pin level change or on alarm.

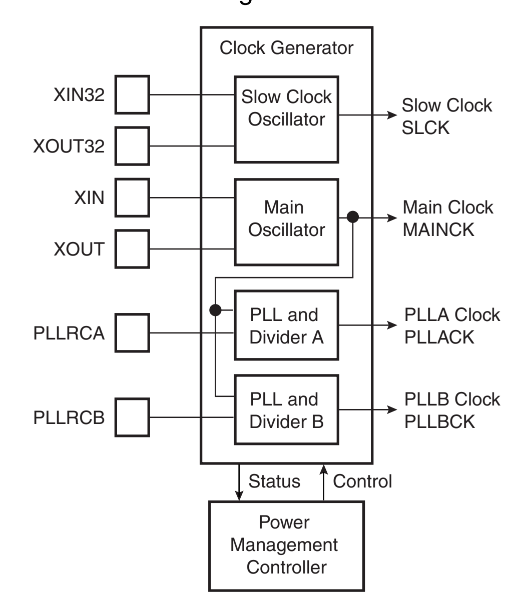
9.4 General-purpose Backup Registers

- Four 32-bit general-purpose backup registers

9.5 Clock Generator

- Embeds the Low-power 32768 Hz Slow Clock Oscillator
  - Provides the permanent Slow Clock to the system
- Embeds the Main Oscillator
  - Oscillator bypass feature
  - Supports 3 to 20 MHz crystals
- Embeds Two PLLs
  - Outputs 80 to 240 MHz clocks
  - Integrates an input divider to increase output accuracy
  - 1 MHz minimum input frequency
- Provides SLCK, MAINCK, PLLACK and PLLBCK.

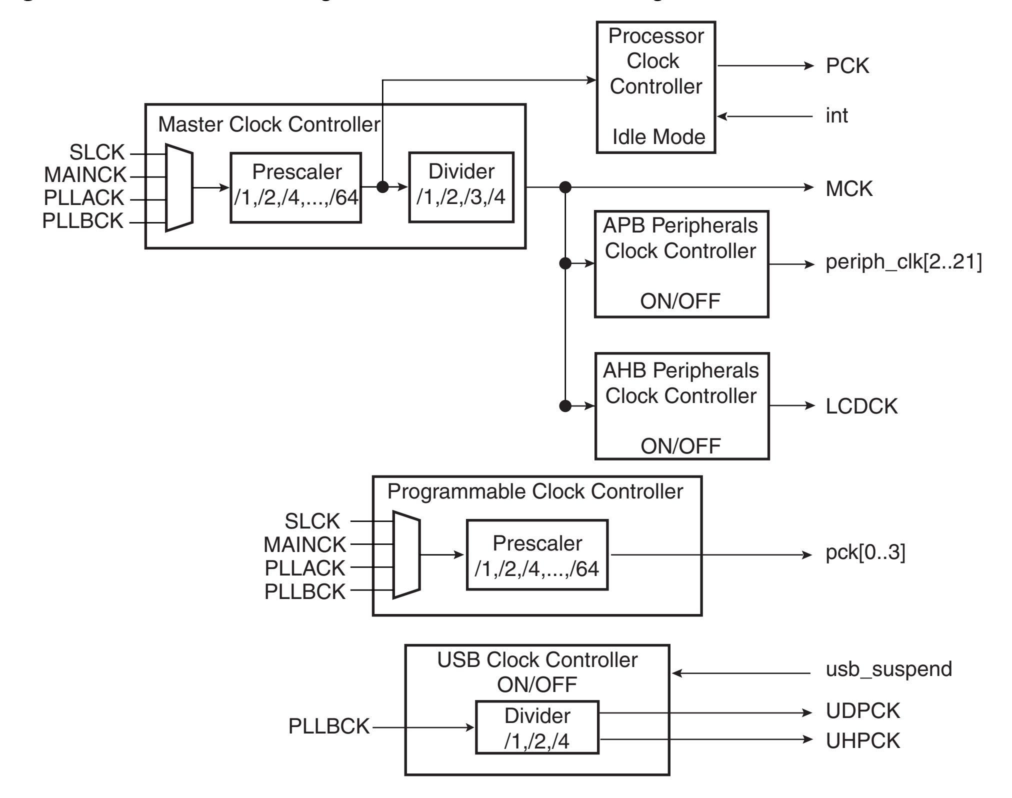
Figure 9-3. Clock Generator Block Diagram



9.6 Power Management Controller

- The Power Management Controller Provides:
  – the Processor Clock PCK
  – the Master Clock MCK
  – the USB Clock USBCK
  – the LCD Controller Clock LCDCK
  – up to thirty peripheral clocks
  – four programmable clock outputs: PCK0 to PCK3

Figure 9-4. Power Management Controller Block Diagram



9.7 Periodic Interval Timer

- Includes a 20-bit Periodic Counter with less than 1 µs accuracy
- Includes a 12-bit Interval Overlay Counter
- Real Time OS or Linux®/Windows CE® compliant tick generator

9.8 Watchdog Timer

- 12-bit key-protected only-once programmable counter
- Windowed, prevents the processor to be in a dead-lock on the watchdog access.

9.9 Real-time Timer

- 32-bit Free-running backup counter
- Alarm Register capable to generate a wake-up of the system

9.10 Advanced Interrupt Controller

- Controls the interrupt lines (nIRQ and nFIQ) of an ARM Processor
- Thirty-two individually maskable and vectored interrupt sources
  - Source 0 is reserved for the Fast Interrupt Input (FIQ)
  - Source 1 is reserved for system peripherals (PIT, RTT, PMC, DBGU, etc.)
  - Source 2 to Source 31 control up to thirty embedded peripheral interrupts or external interrupts
  - Programmable Edge-triggered or Level-sensitive Internal Sources
  - Programmable Positive/Negative Edge-triggered or High/Low Level-sensitive
- Four External Sources
- 8-level Priority Controller
  - Drives the Normal Interrupt of the processor
  - Handles priority of the interrupt sources 1 to 31
  - Higher priority interrupts can be served during service of lower priority interrupt
- Vectoring
  - Optimizes Interrupt Service Routine Branch and Execution
  - One 32-bit Vector Register per interrupt source
  - Interrupt Vector Register reads the corresponding current Interrupt Vector
- Protect Mode
  - Easy debugging by preventing automatic operations when protect mode is enabled
- Fast Forcing
  - Permits redirecting any normal interrupt source on the Fast Interrupt of the processor
- General Interrupt Mask
  - Provides processor synchronization on events without triggering an interrupt

9.11 Debug Unit

- Composed of four functions
  - Two-pin UART
  - Debug Communication Channel (DCC) support
  - Chip ID Registers
  - ICE Access Prevention
- Two-pin UART
  - Implemented features are 100% compatible with the standard Atmel USART
  - Independent receiver and transmitter with a common programmable Baud Rate Generator
  - Even, Odd, Mark or Space Parity Generation
  - Parity, Framing and Overrun Error Detection
  - Automatic Echo, Local Loopback and Remote Loopback Channel Modes

– Support for two PDC channels with connection to receiver and transmitter

- Debug Communication Channel Support
  – Offers visibility of COMMRX and COMMTX signals from the ARM Processor
- Chip ID Registers
  – Identification of the device revision, sizes of the embedded memories, set of peripherals
- ICE Access prevention
  – Enables software to prevent system access through the ARM Processor's ICE
  – Prevention is made by asserting the NTRST line of the ARM Processor's ICE

9.12 PIO Controllers

- Three PIO Controllers, each controlling up to 32 programmable I/O Lines
  – PIOA has 32 I/O Lines
  – PIOB has 32 I/O Lines
  – PIOC has 32 I/O Lines
- Fully programmable through Set/Clear Registers
- Multiplexing of two peripheral functions per I/O Line
- For each I/O Line (whether assigned to a peripheral or used as general-purpose I/O)
  – Input change interrupt
  – Glitch filter
  – Multi-drive option enables driving in open drain
  – Programmable pull up on each I/O line
  – Pin data status register, supplies visibility of the level on the pin at any time
- Synchronous output, provides Set and Clear of several I/O lines in a single write

10. Peripherals

10.1 Peripheral Mapping

10.1.1 User Peripheral Mapping

The User Peripherals are mapped in the upper 256M bytes of the address space between the addresses 0xFFFA 0000 and 0xFFFC FFFF. Each User Peripheral is allocated 16K bytes of address space.

Figure 10-1. User Peripheral Mapping

Start Address	End Address	Peripheral	Peripheral Name	Size
0xFFFA 0000	0xFFFA 3FFF	TC0, TC1, TC2	Timer/Counter 0, 1 and 2	16K Bytes
0xFFFA 4000	0xFFFA 7FFF	UDP	USB Device Port	16K Bytes
0xFFFA 8000	0xFFFA BFFF	MCI	Multimedia Card Interface	16K Bytes
0xFFFA C000	0xFFFA FFFF	TWI	Two-Wire Interface	16K Bytes
0xFFFB 0000	0xFFFB 3FFF	USART0	Universal Synchronous Asynchronous Receiver Transmitter 0	16K Bytes
0xFFFB 4000	0xFFFB 7FFF	USART1	Universal Synchronous Asynchronous Receiver Transmitter 1	16K Bytes
0xFFFB 8000	0xFFFB BFFF	USART2	Universal Synchronous Asynchronous Receiver Transmitter 2	16K Bytes
0xFFFB C000	0xFFFB FFFF	SSC0	Serial Synchronous Controller 0	16K Bytes
0xFFFC 0000	0xFFFC 3FFF	SSC1	Serial Synchronous Controller 1	16K Bytes
0xFFFC 4000	0xFFFC 7FFF	SSC2	Serial Synchronous Controller 2	16K Bytes
0xFFFC 8000	0xFFFC BFFF	SPI0	Serial Peripheral Interface 0	16K Bytes
0xFFFC C000	0xFFFC FFFF	SPI1	Serial Peripheral Interface 1	16K Bytes

10.2 Peripheral Multiplexing on PIO Lines

The AT91SAM9261 features three PIO controllers, PIOA, PIOB and PIOC, that multiplex the I/O lines of the peripheral set.

Each PIO Controller controls up to thirty-two lines. Each line can be assigned to one of two peripheral functions, A or B. The tables in the following pages define how the I/O lines of the peripherals A and B are multiplexed on the PIO Controllers. The two columns “Function” and “Comments” have been inserted for the user’s own comments; they may be used to track how pins are defined in an application.

Note that some output only peripheral functions might be duplicated within the tables.

The column “Reset State” indicates whether the PIO Line resets in I/O mode or in peripheral mode. If I/O is mentioned, the PIO Line resets in input with the pull-up enabled, so that the device is maintained in a static state as soon as the reset is released. As a result, the bit corresponding to the PIO Line in the register PIO_PSR (Peripheral Status Register) resets low.

If a signal name is mentioned in the “Reset State” column, the PIO Line is assigned to this function and the corresponding bit in PIO_PSR resets high. This is the case of pins controlling memories, in particular the address lines, which require the pin to be driven as soon as the reset is released. Note that the pull-up resistor is also enabled in this case.

10.2.1 Resource Multiplexing

10.2.1.1 LCD Controller

The LCD Controller can interface with several LCD panels. It supports 4, 8 or 16 bit-per-pixel without any limitation. Interfacing 24 bit-per-pixel TFTs panel prevents using the SSC0 and using the chip select line 0 of the SPI1.

16 bit-per-pixel TFT panels are interfaced through peripheral B functions, as colors data are output on LCDD3 to LCDD7, LCDD11 to LCDD15 and LCDD19 to LCDD23. Intensity bit is output on LCDD2, LCDD10 and LCDD18. Using the peripheral B does not prevents using the SSC0 and the SPI1 lines.

10.2.1.2 ETM

Using the ETM prevents:

- using the USART1 and USART2 control signals, in particular the SCK lines which are required to use the USART as ISO7816 and the RTS and CTS to handle hardware handshaking on the serial lines. In case the ETM and an ISO7816 connection are both required, the USART0 has to be used as Smart Card interface.
- using the SSC1
- addressing a more than 8-Mbyte static memory, which requires the A23 and A24 address lines
- using the chip select lines 1 to 3 of SPI0 and SPI1

10.2.1.3 EBI

If not required, the NWAIT function (external wait request) can be deactivated by software allowing this pin to be used as a PIO.

10.2.1.4 32-bit Data Bus

Using a 32-bit Data Bus prevents:

- using the three Timer Counter channels' outputs and trigger inputs
- using the SSC2

10.2.1.5 NAND Flash Interface

Using the NAND Flash interface prevents:

- using NCS3, NCS6 and NCS7 to access other parallel devices

10.2.1.6 Compact Flash Interface

Using the CompactFlash interface prevents:

- using NCS4 and/or NCS5 to access other parallel devices

10.2.1.7 SPI0 and the MultiMedia Card Interface

As the DataFlash Card is compatible with the SDCard, it is useful to multiplex SPI and MCI. Here, the SPI0 signal is multiplexed with the MCI.

10.2.1.8 USARTs

- Using the USART1 and USART2 control signals prevents using the ETM.
- Alternatively, using USART0 with its control signals prevents using some clock outputs and interrupt lines.

10.2.1.9 Clock Outputs

- Using the clock outputs multiplexed with the PIO A prevents using the Debug Unit and/or the Two Wire Interface.
- Alternatively, using the second implementation of the clock outputs prevents using the LCD Controller Interface and/or USART0.

10.2.1.10 Interrupt Lines

- Using FIQ prevents using the USART0 control signals.
- Using IRQ0 prevents using the NWAIT EBI signal.
- Using the IRQ1 and/or IRQ2 prevents using the SPI1.



10.2.2 PIO Controller A Multiplexing

Table 10-1. Multiplexing on PIO Controller A

PIO Controller A				Application Usage	
I/O Line	**Peripheral A**	**Peripheral B**	**Reset State**	**Function**	**Comments**
PA0	SPI0_MISO	MCDA0	I/O		
PA1	SPI0_MOSI	MCCDA	I/O		
PA2	SPI0_SPCK	MCCK	I/O		
PA3	SPI0_NPCS0		I/O		
PA4	SPI0_NPCS1	MCDA1	I/O		
PA5	SPI0_NPCS2	MCDA2	I/O		
PA6	SPI0_NPCS3	MCDA3	I/O		
PA7	TWD	PCK0	I/O		
PA8	TWCK	PCK1	I/O		
PA9	DRXD	PCK2	I/O		
PA10	DTXD	PCK3	I/O		
PA11	TSYNC	SCK1	I/O		
PA12	TCLK	RTS1	I/O		
PA13	TPS0	CTS1	I/O		
PA14	TPS1	SCK2	I/O		
PA15	TPS2	RTS2	I/O		
PA16	TPK0	CTS2	I/O		
PA17	TPK1	TF1	I/O		
PA18	TPK2	TK1	I/O		
PA19	TPK3	TD1	I/O		
PA20	TPK4	RD1	I/O		
PA21	TPK5	RK1	I/O		
PA22	TPK6	RF1	I/O		
PA23	TPK7	RTS0	I/O		
PA24	TPK8	SPI1_NPCS1	I/O		
PA25	TPK9	SPI1_NPCS2	I/O		
PA26	TPK10	SPI1_NPCS3	I/O		
PA27	TPK11	SPI0_NPCS1	I/O		
PA28	TPK12	SPI0_NPCS2	I/O		
PA29	TPK13	SPI0_NPCS3	I/O		
PA30	TPK14	A23	A23		
PA31	TPK15	A24	A24		

10.2.3 PIO Controller B Multiplexing

Table 10-2. Multiplexing on PIO Controller B

PIO Controller B				Application Usage	
I/O Line	Peripheral A	Peripheral B	Reset State	Function	Comments
PB0	LCDVSYNC		I/O		
PB1	LCDHSYNC		I/O		
PB2	LCDDOTCK	PCK0	I/O		
PB3	LCDDEN		I/O		
PB4	LCDCC	LCDD2	I/O		
PB5	LCDD0	LCDD3	I/O		
PB6	LCDD1	LCDD4	I/O		
PB7	LCDD2	LCDD5	I/O		
PB8	LCDD3	LCDD6	I/O		
PB9	LCDD4	LCDD7	I/O		
PB10	LCDD5	LCDD10	I/O		
PB11	LCDD6	LCDD11	I/O		
PB12	LCDD7	LCDD12	I/O		
PB13	LCDD8	LCDD13	I/O		
PB14	LCDD9	LCDD14	I/O		
PB15	LCDD10	LCDD15	I/O		
PB16	LCDD11	LCDD19	I/O		
PB17	LCDD12	LCDD20	I/O		
PB18	LCDD13	LCDD21	I/O		
PB19	LCDD14	LCDD22	I/O		
PB20	LCDD15	LCDD23	I/O		
PB21	TF0	LCDD16	I/O		
PB22	TK0	LCDD17	I/O		
PB23	TD0	LCDD18	I/O		
PB24	RD0	LCDD19	I/O		
PB25	RK0	LCDD20	I/O		
PB26	RF0	LCDD21	I/O		
PB27	SPI1_NPCS1	LCDD22	I/O		
PB28	SPI1_NPCS0	LCDD23	I/O		
PB29	SPI1_SPCK	IRQ2	I/O		
PB30	SPI1_MISO	IRQ1	I/O		
PB31	SPI1_MOSI	PCK2	I/O		





10.2.4 PIO Controller C Multiplexing

Table 10-3. Multiplexing on PIO Controller C

PIO Controller C				Application Usage	
I/O Line	**Peripheral A**	**Peripheral B**	**Reset State**	**Function**	**Comments**
PC0	NANDOE	NCS6	I/O		
PC1	NANDWE	NCS7	I/O		
PC2	NWAIT	IRQ0	I/O		
PC3	A25/CFRNW		A25		
PC4	NCS4/CFCS0		I/O		
PC5	NCS5/CFCS1		I/O		
PC6	CFCE1		I/O		
PC7	CFCE2		I/O		
PC8	TXD0	PCK2	I/O		
PC9	RXD0	PCK3	I/O		
PC10	RTS0	SCK0	I/O		
PC11	CTS0	FIQ	I/O		
PC12	TXD1	NCS6	I/O		
PC13	RXD1	NCS7	I/O		
PC14	TXD2	SPI1_NPCS2	I/O		
PC15	RXD2	SPI1_NPCS3	I/O		
PC16	D16	TCLK0	I/O		
PC17	D17	TCLK1	I/O		
PC18	D18	TCLK2	I/O		
PC19	D19	TIOA0	I/O		
PC20	D20	TIOB0	I/O		
PC21	D21	TIOA1	I/O		
PC22	D22	TIOB1	I/O		
PC23	D23	TIOA2	I/O		
PC24	D24	TIOB2	I/O		
PC25	D25	TF2	I/O		
PC26	D26	TK2	I/O		
PC27	D27	TD2	I/O		
PC28	D28	RD2	I/O		
PC29	D29	RK2	I/O		
PC30	D30	RF2	I/O		
PC31	D31	PCK1	I/O		

10.3 Peripheral Identifiers

The AT91SAM9261 embeds a wide range of peripherals. Table 10-4 defines the Peripheral Identifiers of the AT91SAM9261. A peripheral identifier is required for the control of the peripheral interrupt with the Advanced Interrupt Controller and for the control of the peripheral clock with the Power Management Controller.

Table 10-4. Peripheral Identifiers

Peripheral ID	Peripheral Mnemonic	Peripheral Name	External Interrupt
0	AIC	Advanced Interrupt Controller	FIQ
1	SYSIRQ	System Interrupt	
2	PIOA	Parallel I/O Controller A	
3	PIOB	Parallel I/O Controller B	
4	PIOC	Parallel I/O Controller C	
5	-	Reserved	
6	US0	USART 0	
7	US1	USART 1	
8	US2	USART 2	
9	MCI	Multimedia Card Interface	
10	UDP	USB Device Port	
11	TWI	Two-Wire Interface	
12	SPI0	Serial Peripheral Interface 0	
13	SPI1	Serial Peripheral Interface 1	
14	SSC0	Synchronous Serial Controller 0	
15	SSC1	Synchronous Serial Controller 1	
16	SSC2	Synchronous Serial Controller 2	
17	TC0	Timer/Counter 0	
18	TC1	Timer/Counter 1	
19	TC2	Timer/Counter 2	
20	UHP	USB Host Port	
21	LCDC	LCD Controller	
22 - 28	-	Reserved	
29	AIC	Advanced Interrupt Controller	IRQ0
30	AIC	Advanced Interrupt Controller	IRQ1
31	AIC	Advanced Interrupt Controller	IRQ2



10.3.1 System Interrupt

The System Interrupt in Source 1 is the wired-OR of the interrupt signals coming from:

- the SDRAM Controller
- the Debug Unit
- the Periodic Interval Timer
- the Real-Time Timer
- the Watchdog Timer
- the Reset Controller
- the Power Management Controller

The clock of these peripherals cannot be deactivated and Peripheral ID 1 can only be used within the Advanced Interrupt Controller.

10.3.2 External Interrupts

All external interrupt signals, i.e., the Fast Interrupt signal FIQ or the Interrupt signals IRQ0 to IRQ2, use a dedicated Peripheral ID. However, there is no clock control associated with these peripheral IDs.

10.4 External Bus Interface

- Integrates two External Memory Controllers:
  - Static Memory Controller
  - SDRAM Controller
- Additional logic for NAND Flash and CompactFlash support
  - NAND Flash support: 8-bit as well as 16-bit devices are supported
  - CompactFlash support: all modes (Attribute Memory, Common Memory, I/O, True IDE) are supported but the signals -IOIS16 (I/O and True IDE modes) and -ATA SEL (True IDE mode) are not handled.
- Optimized External Bus:
  - 16- or 32-bit Data Bus
  - Up to 26-bit Address Bus, up to 64-Mbytes addressable
  - Eight Chip Selects, each reserved to one of the eight Memory Areas
  - Optimized pin multiplexing to reduce latencies on External Memories
- Configurable Chip Select Assignment Managed by EBI_CSA Register located in the MATRIX user interface:
  - Static Memory Controller on NCS0
  - SDRAM Controller or Static Memory Controller on NCS1
  - Static Memory Controller on NCS2
  - Static Memory Controller on NCS3, Optional NAND Flash Support
  - Static Memory Controller on NCS4 - NCS5, Optional CompactFlash Support
  - Static Memory Controller on NCS6 - NCS7

10.5 Static Memory Controller

- External memory mapping, 256-Mbyte address space per Chip Select Line
- Up to Eight Chip Select Lines
- 8-, 16- or 32-bit Data Bus
- Multiple Access Modes supported
  - Byte Write or Byte Select Lines
  - Asynchronous read in Page Mode supported (4- up to 32-byte page size)
- Multiple device adaptability
  - Compliant with LCD Module
  - Control signal programmable setup, pulse and hold time for each Memory Bank
- Multiple Wait State Management
  - Programmable Wait State Generation
  - External Wait Request
  - Programmable Data Float Time
- Slow Clock Mode Supported

10.6 SDRAM Controller

- Supported Devices:
  - Standard and Low Power SDRAM (Mobile SDRAM)
- Numerous configurations supported
  - 2K, 4K, 8K Row Address Memory Parts
  - SDRAM with two or four Internal Banks
  - SDRAM with 16- or 32-bit Data Path
- Programming Facilities
  - Word, half-word, byte access
  - Automatic page break when Memory Boundary has been reached
  - Multibank Ping-pong Access
  - Timing parameters specified by software
  - Automatic refresh operation, refresh rate is programmable
- Energy-saving Capabilities
  - Self-refresh, power down and deep power down modes supported
- Error detection
  - Refresh Error Interrupt
- SDRAM Power-up Initialization by software
- CAS Latency of 1, 2 and 3 supported
- Auto Precharge Command not used



10.7 Serial Peripheral Interface

- Supports communication with serial external devices
  - Four chip selects with external decoder support allow communication with up to fifteen peripherals
  - Serial memories, such as DataFlash and 3-wire EEPROMs
  - Serial peripherals, such as ADCs, DACs, LCD Controllers, CAN Controllers and Sensors
  - External co-processors
- Master or slave serial peripheral bus interface
  - 8- to 16-bit programmable data length per chip select
  - Programmable phase and polarity per chip select
  - Programmable transfer delays between consecutive transfers and between clock and data per chip select
  - Programmable delay between consecutive transfers
  - Selectable mode fault detection
- Very fast transfers supported
  - Transfers with baud rates up to MCK
  - The chip select line may be left active to speed up transfers on the same device

10.8 Two-wire Interface

- Compatibility with standard two-wire serial memory
- One, two or three bytes for slave address
- Sequential read/write operations

10.9 USART

- Programmable Baud Rate Generator
- 5- to 9-bit full-duplex synchronous or asynchronous serial communications
  - 1, 1.5 or 2 stop bits in Asynchronous Mode or 1 or 2 stop bits in Synchronous Mode
  - Parity generation and error detection
  - Framing error detection, overrun error detection
  - MSB- or LSB-first
  - Optional break generation and detection
  - By-8 or by-16 over-sampling receiver frequency
  - Hardware handshaking RTS-CTS
  - Receiver time-out and transmitter timeguard
  - Optional Multi-drop Mode with address generation and detection
  - Optional Manchester Encoding
- RS485 with driver control signal
- ISO7816, T = 0 or T = 1 Protocols for interfacing with smart cards
  - NACK handling, error counter with repetition and iteration limit

- IrDA modulation and demodulation
  - Communication at up to 115.2 Kbps
- Test Modes
  - Remote Loopback, Local Loopback, Automatic Echo

10.10 Synchronous Serial Controller

- Provides serial synchronous communication links used in audio and telecom applications (with CODECs in Master or Slave Modes, I^{2}S, TDM Buses, Magnetic Card Reader and more).
- Contains an independent receiver and transmitter and a common clock divider.
- Offers a configurable frame sync and data length.
- Receiver and transmitter can be programmed to start automatically or on detection of different event on the frame sync signal.
- Receiver and transmitter include a data signal, a clock signal and a frame synchronization signal.

10.11 Timer Counter

- Three 16-bit Timer Counter Channels
- Wide range of functions including:
  - Frequency Measurement
  - Event Counting
  - Interval Measurement
  - Pulse Generation
  - Delay Timing
  - Pulse Width Modulation
  - Up/down Capabilities
- Each channel is user-configurable and contains:
  - Three external clock inputs
  - Five internal clock inputs
  - Two multi-purpose input/output signals
- Two global registers that act on all three TC Channels
- As mentioned above, the three Timer Counter Channels (0 to 2) are described with five generic clock inputs, named TIMER_CLOCK1 to TIMER_CLOCK5. In the AT91SAM9261, these clock inputs are connected to the Master Clock (MCK), to the Slow Clock (SLCK) and to divisions of the Master Clock.
  Table 10-5 gives the assignment of these Timer Counter clock inputs in the AT91SAM9261, which is common to the Timer Counter 0 to 2.



Table 10-5. Timer Counter Clock Assignment

TC Clock input	Clock
TIMER_CLOCK1	MCK/2
TIMER_CLOCK2	MCK/8
TIMER_CLOCK3	MCK/32
TIMER_CLOCK4	MCK/128
TIMER_CLOCK5	SLCK

10.12 Multimedia Card Interface

- Compatibility with MultiMedia Card Specification Version 2.2
- Compatibility with SD Memory Card Specification Version 1.0
- Compatibility with SDIO Specification Version V1.0
- Cards clock rate up to Master Clock divided by 2
- Embedded power management to slow down clock rate when not used
- Each MCI has two slots, each supporting
  - One slot for one MultiMedia Card bus (up to 30 cards) or
  - One SD Memory Card
- Support for stream, block and multi-block data read and write

10.13 USB

- USB Host Port:
  - Compliance with Open HCI Rev 1.0 specification
  - Compliance with USB V2.0 Full-speed and Low-speed Specification
  - Supports both Low-speed 1.5 Mbps and Full-speed 12 Mbps USB devices
  - Root hub integrated with two downstream USB ports
  - Two embedded USB transceivers
  - No overcurrent detection
  - Supports power management
  - Operates as a master on the Bus Matrix
- USB Device Port:
  - USB V2.0 full-speed compliant, 12 Mbits per second
  - Embedded USB V2.0 full-speed transceiver
  - Embedded dual-port RAM for endpoints
  - Suspend/Resume logic
  - Ping-pong mode (two memory banks) for isochronous and bulk endpoints.

- Six general-purpose endpoints:
  - Endpoint 0: 8 bytes, no ping-pong mode
  - Endpoint 1, Endpoint 2: 64 bytes, ping-pong mode
  - Endpoint 3: 64 bytes, no ping-pong mode
  - Endpoint 4, Endpoint 5: 256 bytes, ping-pong mode
- Embedded pad pull-up configurable via USB_PUCR Register located in the MATRIX user interface

10.14 LCD Controller

- Single and Dual scan color and monochrome passive STN LCD panels supported
- Single scan active TFT LCD panels supported.
- 4-bit single scan, 8-bit single or dual scan, 16-bit dual scan STN interfaces supported
- Up to 24-bit single scan TFT interfaces supported
- Up to 16 gray levels for mono STN and up to 4096 colors for color STN displays
- 1, 2 bits per pixel (palletized), 4 bits per pixel (non-palletized) for mono STN
- 1, 2, 4, 8 bits per pixel (palletized), 16 bits per pixel (non-palletized) for color STN
- 1, 2, 4, 8 bits per pixel (palletized), 16, 24 bits per pixel (non-palletized) for TFT
- Single clock domain architecture
- Resolution supported up to 2048 x 2048

11. ARM926EJ-S Processor Overview

11.1 Overview

The ARM926EJ-S processor is a member of the ARM9™ family of general-purpose microprocessors. The ARM926EJ-S implements ARM architecture version 5TEJ and is targeted at multi-tasking applications where full memory management, high performance, low die size and low power are all important features.

The ARM926EJ-S processor supports the 32-bit ARM and 16-bit THUMB instruction sets, enabling the user to trade off between high performance and high code density. It also supports 8-bit Java instruction set and includes features for efficient execution of Java bytecode, providing a Java performance similar to a JIT (just-In-Time compilers), for the next generation of Java-powered wireless and embedded devices. It includes an enhanced multiplier design for improved DSP performance.

The ARM926EJ-S processor supports the ARM debug architecture and includes logic to assist in both hardware and software debug.

The ARM926EJ-S provides a complete high performance processor subsystem, including:

- an ARM9EJ-S integer core
- a Memory Management Unit (MMU)
- separate instruction and data AMBA™ AHB bus interfaces
- separate instruction and data TCM interfaces

11.2 Block Diagram

Figure 11-1. ARM926EJ-S Internal Functional Block Diagram

11.3 ARM9EJ-S Processor

11.3.1 ARM9EJ-S™ Operating States

The ARM9EJ-S processor can operate in three different states, each with a specific instruction set:

- ARM state: 32-bit, word-aligned ARM instructions.
- THUMB state: 16-bit, halfword-aligned Thumb instructions.
- Jazelle state: variable length, byte-aligned Jazelle instructions.

In Jazelle state, all instruction Fetches are in words.

11.3.2 Switching State

The operating state of the ARM9EJ-S core can be switched between:

- ARM state and THUMB state using the BX and BLX instructions, and loads to the PC

- ARM state and Jazelle state using the BXJ instruction

All exceptions are entered, handled and exited in ARM state. If an exception occurs in Thumb or Jazelle states, the processor reverts to ARM state. The transition back to Thumb or Jazelle states occurs automatically on return from the exception handler.

11.3.3 Instruction Pipelines

The ARM9EJ-S core uses two kinds of pipelines to increase the speed of the flow of instructions to the processor.

A five-stage (five clock cycles) pipeline is used for ARM and Thumb states. It consists of Fetch, Decode, Execute, Memory and Writeback stages.

A six-stage (six clock cycles) pipeline is used for Jazelle state It consists of Fetch, Jazelle/Decode (two clock cycles), Execute, Memory and Writeback stages.

11.3.4 Memory Access

The ARM9EJ-S core supports byte (8-bit), half-word (16-bit) and word (32-bit) access. Words must be aligned to four-byte boundaries, half-words must be aligned to two-byte boundaries and bytes can be placed on any byte boundary.

Because of the nature of the pipelines, it is possible for a value to be required for use before it has been placed in the register bank by the actions of an earlier instruction. The ARM9EJ-S control logic automatically detects these cases and stalls the core or forward data.

11.3.5 Jazelle Technology

The Jazelle technology enables direct and efficient execution of Java byte codes on ARM processors, providing high performance for the next generation of Java-powered wireless and embedded devices.

The new Java feature of ARM9EJ-S can be described as a hardware emulation of a JVM (Java Virtual Machine). Java mode will appear as another state: instead of executing ARM or Thumb instructions, it executes Java byte codes. The Java byte code decoder logic implemented in ARM9EJ-S decodes 95% of executed byte codes and turns them into ARM instructions without any overhead, while less frequently used byte codes are broken down into optimized sequences of ARM instructions. The hardware/software split is invisible to the programmer, invisible to the application and invisible to the operating system. All existing ARM registers are re-used in Jazelle state and all registers then have particular functions in this mode.

Minimum interrupt latency is maintained across both ARM state and Java state. Since byte codes execution can be restarted, an interrupt automatically triggers the core to switch from Java state to ARM state for the execution of the interrupt handler. This means that no special provision has to be made for handling interrupts while executing byte codes, whether in hardware or in software.

11.3.6 ARM9EJ-S Operating Modes

In all states, there are seven operation modes:

- User mode is the usual ARM program execution state. It is used for executing most application programs
- Fast Interrupt (FIQ) mode is used for handling fast interrupts. It is suitable for high-speed data transfer or channel process

- Interrupt (IRQ) mode is used for general-purpose interrupt handling
- Supervisor mode is a protected mode for the operating system
- Abort mode is entered after a data or instruction prefetch abort
- System mode is a privileged user mode for the operating system
- Undefined mode is entered when an undefined instruction exception occurs

Mode changes may be made under software control, or may be brought about by external interrupts or exception processing. Most application programs execute in User Mode. The non-user modes, known as privileged modes, are entered in order to service interrupts or exceptions or to access protected resources.

11.3.7 ARM9EJ-S Registers

The ARM9EJ-S core has a total of 37 registers.

- 31 general-purpose 32-bit registers
- 6 32-bit status registers

Table 11-1 shows all the registers in all modes.

Table 11-1. ARM9EJ-S® Modes and Registers Layout

User and System Mode	Supervisor Mode	Abort Mode	Undefined Mode	Interrupt Mode	Fast Interrupt Mode
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8_FIQ
R9	R9	R9	R9	R9	R9_FIQ
R10	R10	R10	R10	R10	R10_FIQ
R11	R11	R11	R11	R11	R11_FIQ
R12	R12	R12	R12	R12	R12_FIQ
R13	R13_SVC	R13_ABORT	R13_UNDEF	R13_IRQ	R13_FIQ
R14	R14_SVC	R14_ABORT	R14_UNDEF	R14_IRQ	R14_FIQ
PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR

Table 11-1. ARM9EJ-S® Modes and Registers Layout

User and System Mode	Supervisor Mode	Abort Mode	Undefined Mode	Interrupt Mode	Fast Interrupt Mode
	SPSR_SVC	SPSR_ABORT	SPSR_UNDEF	SPSR_IRQ	SPSR_FIQ

Mode-specific banked registers

The ARM state register set contains 16 directly-accessible registers, r0 to r15, and an additional register, the Current Program Status Register (CPSR). Registers r0 to r13 are general-purpose registers used to hold either data or address values. Register r14 is used as a Link register that holds a value (return address) of r15 when BL or BLX is executed. Register r15 is used as a program counter (PC), whereas the Current Program Status Register (CPSR) contains condition code flags and the current mode bits.

In privileged modes (FIQ, Supervisor, Abort, IRQ, Undefined), mode-specific banked registers (r8 to r14 in FIQ mode or r13 to r14 in the other modes) become available. The corresponding banked registers r14_fiq, r14_svc, r14_abt, r14_irq, r14_und are similarly used to hold the values (return address for each mode) of r15 (PC) when interrupts and exceptions arise, or when BL or BLX instructions are executed within interrupt or exception routines. There is another register called Saved Program Status Register (SPSR) that becomes available in privileged modes instead of CPSR. This register contains condition code flags and the current mode bits saved as a result of the exception that caused entry to the current (privileged) mode.

In all modes and due to a software agreement, register r13 is used as stack pointer.

The use and the function of all the registers described above should obey ARM Procedure Call Standard (APCS) which defines:

- constraints on the use of registers
- stack conventions
- argument passing and result return

The Thumb state register set is a subset of the ARM state set. The programmer has direct access to:

- Eight general-purpose registers r0-r7
- Stack pointer, SP
- Link register, LR (ARM r14)
- PC
- CPSR

There are banked registers SPs, LRs and SPSRs for each privileged mode (for more details see the ARM9EJ-S Technical Reference Manual, ref. DDI0222B, revision r1p2 page 2-12).

11.3.7.1 Status Registers

The ARM9EJ-S core contains one CPSR, and five SPSRs for exception handlers to use. The program status registers:

- hold information about the most recently performed ALU operation

- control the enabling and disabling of interrupts
- set the processor operation mode

Figure 11-2. Status Register Format

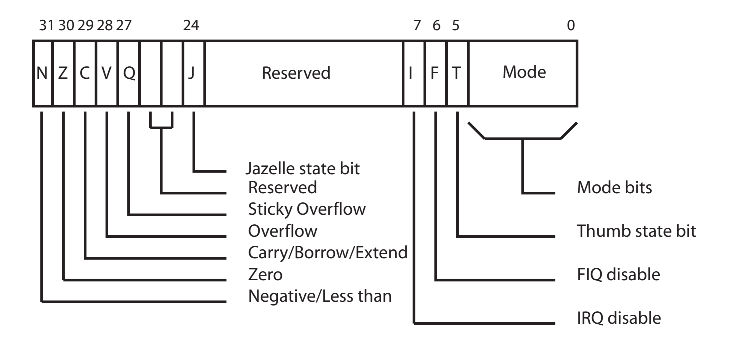


Figure 11-2 shows the status register format, where:

- N: Negative, Z: Zero, C: Carry, and V: Overflow are the four ALU flags
- The Sticky Overflow (Q) flag can be set by certain multiply and fractional arithmetic instructions like QADD, QDADD, QSUB, QDSUB, SMLAxy, and SMLAWy needed to achieve DSP operations.
  The Q flag is sticky in that, when set by an instruction, it remains set until explicitly cleared by an MSR instruction writing to the CPSR. Instructions cannot execute conditionally on the status of the Q flag.
- The J bit in the CPSR indicates when the ARM9EJ-S core is in Jazelle state, where:
  – J = 0: The processor is in ARM or Thumb state, depending on the T bit
  – J = 1: The processor is in Jazelle state.
- Mode: five bits to encode the current processor mode

11.3.7.2 Exceptions

Exception Types and Priorities

The ARM9EJ-S supports five types of exceptions. Each type drives the ARM9EJ-S in a privileged mode. The types of exceptions are:

- Fast interrupt (FIQ)
- Normal interrupt (IRQ)
- Data and Prefetched aborts (Abort)
- Undefined instruction (Undefined)
- Software interrupt and Reset (Supervisor)

When an exception occurs, the banked version of R14 and the SPSR for the exception mode are used to save the state.

More than one exception can happen at a time, therefore the ARM9EJ-S takes the arisen exceptions according to the following priority order:

- Reset (highest priority)
- Data Abort
- FIQ
- IRQ



- Prefetch Abort
- BKPT, Undefined instruction, and Software Interrupt (SWI) (Lowest priority)

The BKPT, or Undefined instruction, and SWI exceptions are mutually exclusive.

There is one exception in the priority scheme though, when FIQs are enabled and a Data Abort occurs at the same time as an FIQ, the ARM9EJ-S core enters the Data Abort handler, and proceeds immediately to FIQ vector. A normal return from the FIQ causes the Data Abort handler to resume execution. Data Aborts must have higher priority than FIQs to ensure that the transfer error does not escape detection.

Exception Modes and Handling

Exceptions arise whenever the normal flow of a program must be halted temporarily, for example, to service an interrupt from a peripheral.

When handling an ARM exception, the ARM9EJ-S core performs the following operations:

1. Preserves the address of the next instruction in the appropriate Link Register that corresponds to the new mode that has been entered. When the exception entry is from:
   – ARM and Jazelle states, the ARM9EJ-S copies the address of the next instruction into LR (current PC(r15) + 4 or PC + 8 depending on the exception).
   – THUMB state, the ARM9EJ-S writes the value of the PC into LR, offset by a value (current PC + 2, PC + 4 or PC + 8 depending on the exception) that causes the program to resume from the correct place on return.
2. Copies the CPSR into the appropriate SPSR.
3. Forces the CPSR mode bits to a value that depends on the exception.
4. Forces the PC to fetch the next instruction from the relevant exception vector.

The register r13 is also banked across exception modes to provide each exception handler with private stack pointer.

The ARM9EJ-S can also set the interrupt disable flags to prevent otherwise unmanageable nesting of exceptions.

When an exception has completed, the exception handler must move both the return value in the banked LR minus an offset to the PC and the SPSR to the CPSR. The offset value varies according to the type of exception. This action restores both PC and the CPSR.

The fast interrupt mode has seven private registers r8 to r14 (banked registers) to reduce or remove the requirement for register saving which minimizes the overhead of context switching.

The Prefetch Abort is one of the aborts that indicates that the current memory access cannot be completed. When a Prefetch Abort occurs, the ARM9EJ-S marks the prefetched instruction as invalid, but does not take the exception until the instruction reaches the Execute stage in the pipeline. If the instruction is not executed, for example because a branch occurs while it is in the pipeline, the abort does not take place.

The breakpoint (BKPT) instruction is a new feature of ARM9EJ-S that is destined to solve the problem of the Prefetch Abort. A breakpoint instruction operates as though the instruction caused a Prefetch Abort.

A breakpoint instruction does not cause the ARM9EJ-S to take the Prefetch Abort exception until the instruction reaches the Execute stage of the pipeline. If the instruction is not executed, for example because a branch occurs while it is in the pipeline, the breakpoint does not take place.

11.3.8 ARM Instruction Set Overview

The ARM instruction set is divided into:

- Branch instructions
- Data processing instructions
- Status register transfer instructions
- Load and Store instructions
- Coprocessor instructions
- Exception-generating instructions

ARM instructions can be executed conditionally. Every instruction contains a 4-bit condition code field (bits[31:28]).

Table 11-2 gives the ARM instruction mnemonic list.

Table 11-2. ARM Instruction Mnemonic List

Mnemonic	Operation	Mnemonic	Operation
MOV	Move	MVN	Move Not
ADD	Add	ADC	Add with Carry
SUB	Subtract	SBC	Subtract with Carry
RSB	Reverse Subtract	RSC	Reverse Subtract with Carry
CMP	Compare	CMN	Compare Negated
TST	Test	TEQ	Test Equivalence
AND	Logical AND	BIC	Bit Clear
EOR	Logical Exclusive OR	ORR	Logical (inclusive) OR
MUL	Multiply	MLA	Multiply Accumulate
SMULL	Sign Long Multiply	UMULL	Unsigned Long Multiply
SMLAL	Signed Long Multiply Accumulate	UMLAL	Unsigned Long Multiply Accumulate
MSR	Move to Status Register	MRS	Move From Status Register
B	Branch	BL	Branch and Link
BX	Branch and Exchange	SWI	Software Interrupt
LDR	Load Word	STR	Store Word
LDRSH	Load Signed Halfword		
LDRSB	Load Signed Byte		
LDRH	Load Half Word	STRH	Store Half Word
LDRB	Load Byte	STRB	Store Byte
LDRBT	Load Register Byte with Translation	STRBT	Store Register Byte with Translation
LDRT	Load Register with Translation	STRT	Store Register with Translation
LDM	Load Multiple	STM	Store Multiple
SWP	Swap Word	SWPB	Swap Byte

Table 11-2. ARM Instruction Mnemonic List (Continued)

Mnemonic	Operation	Mnemonic	Operation
MCR	Move To Coprocessor	MRC	Move From Coprocessor
LDC	Load To Coprocessor	STC	Store From Coprocessor
CDP	Coprocessor Data Processing		

11.3.9 New ARM Instruction Set

.

Table 11-3. New ARM Instruction Mnemonic List

Mnemonic	Operation	Mnemonic	Operation
BXJ	Branch and exchange to Java	MRRC	Move double from coprocessor
BLX [(1)]	Branch, Link and exchange	MCR2	Alternative move of ARM reg to coprocessor
SMLAxy	Signed Multiply Accumulate 16 * 16 bit	MCRR	Move double to coprocessor
SMLAL	Signed Multiply Accumulate Long	CDP2	Alternative Coprocessor Data Processing
SMLAWy	Signed Multiply Accumulate 32 * 16 bit	BKPT	Breakpoint
SMULxy	Signed Multiply 16 * 16 bit	PLD	Soft Preload, Memory prepare to load from address
SMULWy	Signed Multiply 32 * 16 bit	STRD	Store Double
QADD	Saturated Add	STC2	Alternative Store from Coprocessor
QDADD	Saturated Add with Double	LDRD	Load Double
QSUB	Saturated subtract	LDC2	Alternative Load to Coprocessor
QDSUB	Saturated Subtract with double	CLZ	Count Leading Zeroes

Notes: 1. A Thumb BLX contains two consecutive Thumb instructions, and takes four cycles.

11.3.10 Thumb Instruction Set Overview

The Thumb instruction set is a re-encoded subset of the ARM instruction set.

The Thumb instruction set is divided into:

- Branch instructions
- Data processing instructions
- Load and Store instructions
- Load and Store multiple instructions
- Exception-generating instruction

Table 5 shows the Thumb instruction set. Table 11-4 gives the Thumb instruction mnemonic list.

Table 11-4. Thumb Instruction Mnemonic List

Mnemonic	Operation	Mnemonic	Operation
MOV	Move	MVN	Move Not
ADD	Add	ADC	Add with Carry
SUB	Subtract	SBC	Subtract with Carry
CMP	Compare	CMN	Compare Negated
TST	Test	NEG	Negate
AND	Logical AND	BIC	Bit Clear
EOR	Logical Exclusive OR	ORR	Logical (inclusive) OR
LSL	Logical Shift Left	LSR	Logical Shift Right
ASR	Arithmetic Shift Right	ROR	Rotate Right
MUL	Multiply	BLX	Branch, Link, and Exchange
B	Branch	BL	Branch and Link
BX	Branch and Exchange	SWI	Software Interrupt
LDR	Load Word	STR	Store Word
LDRH	Load Half Word	STRH	Store Half Word
LDRB	Load Byte	STRB	Store Byte
LDRSH	Load Signed Halfword	LDRSB	Load Signed Byte
LDMIA	Load Multiple	STMIA	Store Multiple
PUSH	Push Register to stack	POP	Pop Register from stack
BCC	Conditional Branch	BKPT	Breakpoint

11.4 CP15 Coprocessor

Coprocessor 15, or System Control Coprocessor CP15, is used to configure and control all the items in the list below:

- ARM9EJ-S
- Caches (ICache, DCache and write buffer)
- TCM
- MMU
- Other system options

To control these features, CP15 provides 16 additional registers. See Table 11-5.

Table 11-5. CP15 Registers

Register	Name	Read/Write
0	ID Code[(1)]	Read/Unpredictable
0	Cache type[(1)]	Read/Unpredictable
0	TCM status[(1)]	Read/Unpredictable
1	Control	Read/write
2	Translation Table Base	Read/write
3	Domain Access Control	Read/write
4	Reserved	None
5	Data fault Status[(1)]	Read/write
5	Instruction fault status[(1)]	Read/write
6	Fault Address	Read/write
7	Cache Operations	Read/Write
8	TLB operations	Unpredictable/Write
9	cache lockdown[(2)]	Read/write
9	TCM region	Read/write
10	TLB lockdown	Read/write
11	Reserved	None
12	Reserved	None
13	FCSE PID[(1)]	Read/write
13	Context ID[(1)]	Read/Write
14	Reserved	None
15	Test configuration	Read/Write

Notes: 1. Register locations 0,5, and 13 each provide access to more than one register. The register accessed depends on the value of the opcode_2 field.

2. Register location 9 provides access to more than one register. The register accessed depends on the value of the CRm field.

11.4.1 CP15 Registers Access

CP15 registers can only be accessed in privileged mode by:

- MCR (Move to Coprocessor from ARM Register) instruction is used to write an ARM register to CP15.
- MRC (Move to ARM Register from Coprocessor) instruction is used to read the value of CP15 to an ARM register.

Other instructions like CDP, LDC, STC can cause an undefined instruction exception.

The assembler code for these instructions is:

```
MCR/MRC{cond} p15, opcode_1, Rd, CRn, CRm, opcode_2.
```

The MCR, MRC instructions bit pattern is shown below:

31	30	29	28	27	26	25	24
cond				1	1	1	0

23	22	21	20	19	18	17	16
opcode_1			L	CRn			

15	14	13	12	11	10	9	8
Rd				1	1	1	1

7	6	5	4	3	2	1	0
opcode_2			1	CRm			

- **CRm[3:0]: Specified Coprocessor Action**

Determines specific coprocessor action. Its value is dependent on the CP15 register used. For details, refer to CP15 specific register behavior.

- **opcode_2[7:5]**

Determines specific coprocessor operation code. By default, set to 0.

- **Rd[15:12]: ARM Register**

Defines the ARM register whose value is transferred to the coprocessor. If R15 is chosen, the result is unpredictable.

- **CRn[19:16]: Coprocessor Register**

Determines the destination coprocessor register.

- **L: Instruction Bit**

0 = MCR instruction

1 = MRC instruction

- **opcode_1[23:20]: Coprocessor Code**

Defines the coprocessor specific code. Value is c15 for CP15.

- **cond [31:28]: Condition**

For more details, see Chapter 2 in ARM926EJ-S TRM, ref. DDI0198B.



11.5 Memory Management Unit (MMU)

The ARM926EJ-S processor implements an enhanced ARM architecture v5 MMU to provide virtual memory features required by operating systems like Symbian OS, WindowsCE, and Linux. These virtual memory features are memory access permission controls and virtual to physical address translations.

The Virtual Address generated by the CPU core is converted to a Modified Virtual Address (MVA) by the FCSE (Fast Context Switch Extension) using the value in CP15 register13. The MMU translates modified virtual addresses to physical addresses by using a single, two-level page table set stored in physical memory. Each entry in the set contains the access permissions and the physical address that correspond to the virtual address.

The first level translation tables contain 4096 entries indexed by bits [31:20] of the MVA. These entries contain a pointer to either a 1 MB section of physical memory along with attribute information (access permissions, domain, etc.) or an entry in the second level translation tables; coarse table and fine table.

The second level translation tables contain two subtables, coarse table and fine table. An entry in the coarse table contains a pointer to both large pages and small pages along with access permissions. An entry in the fine table contains a pointer to large, small and tiny pages.

Table 7 shows the different attributes of each page in the physical memory.

Table 11-6. Mapping Details

Mapping Name	Mapping Size	Access Permission By	Subpage Size
Section	1M byte	Section	-
Large Page	64K bytes	4 separated subpages	16K bytes
Small Page	4K bytes	4 separated subpages	1K byte
Tiny Page	1K byte	Tiny Page	-

The MMU consists of:

- Access control logic
- Translation Look-aside Buffer (TLB)
- Translation table walk hardware

11.5.1 Access Control Logic

The access control logic controls access information for every entry in the translation table. The access control logic checks two pieces of access information: domain and access permissions. The domain is the primary access control mechanism for a memory region; there are 16 of them. It defines the conditions necessary for an access to proceed. The domain determines whether the access permissions are used to qualify the access or whether they should be ignored.

The second access control mechanism is access permissions that are defined for sections and for large, small and tiny pages. Sections and tiny pages have a single set of access permissions whereas large and small pages can be associated with 4 sets of access permissions, one for each subpage (quarter of a page).

11.5.2 Translation Look-aside Buffer (TLB)

The Translation Look-aside Buffer (TLB) caches translated entries and thus avoids going through the translation process every time. When the TLB contains an entry for the MVA (Modified Virtual Address), the access control logic determines if the access is permitted and outputs the appropriate physical address corresponding to the MVA. If access is not permitted, the MMU signals the CPU core to abort.

If the TLB does not contain an entry for the MVA, the translation table walk hardware is invoked to retrieve the translation information from the translation table in physical memory.

11.5.3 Translation Table Walk Hardware

The translation table walk hardware is a logic that traverses the translation tables located in physical memory, gets the physical address and access permissions and updates the TLB.

The number of stages in the hardware table walking is one or two depending whether the address is marked as a section-mapped access or a page-mapped access.

There are three sizes of page-mapped accesses and one size of section-mapped access. Page-mapped accesses are for large pages, small pages and tiny pages. The translation process always begins with a level one fetch. A section-mapped access requires only a level one fetch, but a page-mapped access requires an additional level two fetch. For further details on the MMU, please refer to chapter 3 in ARM926EJ-S Technical Reference Manual, ref. DDI0198B.

11.5.4 MMU Faults

The MMU generates an abort on the following types of faults:

- Alignment faults (for data accesses only)
- Translation faults
- Domain faults
- Permission faults

The access control mechanism of the MMU detects the conditions that produce these faults. If the fault is a result of memory access, the MMU aborts the access and signals the fault to the CPU core.The MMU retains status and address information about faults generated by the data accesses in the data fault status register and fault address register. It also retains the status of faults generated by instruction fetches in the instruction fault status register.

The fault status register (register 5 in CP15) indicates the cause of a data or prefetch abort, and the domain number of the aborted access when it happens. The fault address register (register 6 in CP15) holds the MVA associated with the access that caused the Data Abort. For further details on MMU faults, please refer to chapter 3 in ARM926EJ-S Technical Reference Manual, ref. DDI0198B.

11.6 Caches and Write Buffer

The ARM926EJ-S contains a 16 KB Instruction Cache (ICache), a 16 KB Data Cache (DCache), and a write buffer. Although the ICache and DCache share common features, each still has some specific mechanisms.

The caches (ICache and DCache) are four-way set associative, addressed, indexed and tagged using the Modified Virtual Address (MVA), with a cache line length of eight words with two dirty bits for the DCache. The ICache and DCache provide mechanisms for cache lock-down, cache pollution control, and line replacement.



A new feature is now supported by ARM926EJ-S caches called allocate on read-miss commonly known as wrapping. This feature enables the caches to perform critical word first cache refilling. This means that when a request for a word causes a read-miss, the cache performs an AHB access. Instead of loading the whole line (eight words), the cache loads the critical word first, so the processor can reach it quickly, and then the remaining words, no matter where the word is located in the line.

The caches and the write buffer are controlled by the CP15 register 1 (Control), CP15 register 7 (cache operations) and CP15 register 9 (cache lockdown).

11.6.1 Instruction Cache (ICache)

The ICache caches fetched instructions to be executed by the processor. The ICache can be enabled by writing 1 to I bit of the CP15 Register 1 and disabled by writing 0 to this same bit.

When the MMU is enabled, all instruction fetches are subject to translation and permission checks. If the MMU is disabled, all instructions fetches are cachable, no protection checks are made and the physical address is flat-mapped to the modified virtual address. With the MVA use disabled, context switching incurs ICache cleaning and/or invalidating.

When the ICache is disabled, all instruction fetches appear on external memory (AHB) (see Tables 4-1 and 4-2 in page 4-4 in ARM926EJ-S TRM, ref. DDI0198B).

On reset, the ICache entries are invalidated and the ICache is disabled. For best performance, ICache should be enabled as soon as possible after reset.

11.6.2 Data Cache (DCache) and Write Buffer

ARM926EJ-S includes a DCache and a write buffer to reduce the effect of main memory bandwidth and latency on data access performance. The operations of DCache and write buffer are closely connected.

11.6.2.1 DCache

The DCache needs the MMU to be enabled. All data accesses are subject to MMU permission and translation checks. Data accesses that are aborted by the MMU do not cause linefills or data accesses to appear on the AMBA ASB interface. If the MMU is disabled, all data accesses are noncachable, nonbufferable, with no protection checks, and appear on the AHB bus. All addresses are flat-mapped, VA = MVA = PA, which incurs DCache cleaning and/or invalidating every time a context switch occurs.

The DCache stores the Physical Address Tag (PA Tag) from which every line was loaded and uses it when writing modified lines back to external memory. This means that the MMU is not involved in write-back operations.

Each line (8 words) in the DCache has two dirty bits, one for the first four words and the other one for the second four words. These bits, if set, mark the associated half-lines as dirty. If the cache line is replaced due to a linefill or a cache clean operation, the dirty bits are used to decide whether all, half or none is written back to memory.

DCache can be enabled or disabled by writing either 1 or 0 to bit C in register 1 of CP15 (see Tables 4-3 and 4-4 on page 4-5 in ARM926EJ-S TRM, ref. DDI0222B).

The DCache supports write-through and write-back cache operations, selected by memory region using the C and B bits in the MMU translation tables.

The DCache contains an eight data word entry, single address entry write-back buffer used to hold write-back data for cache line eviction or cleaning of dirty cache lines.

The Write Buffer can hold up to 16 words of data and four separate addresses. DCache and Write Buffer operations are closely connected as their configuration is set in each section by the page descriptor in the MMU translation table.

11.6.2.2 Write Buffer

The ARM926EJ-S contains a write buffer that has a 16-word data buffer and a four- address buffer. The write buffer is used for all writes to a bufferable region, write-through region and write-back region. It also allows to avoid stalling the processor when writes to external memory are performed. When a store occurs, data is written to the write buffer at core speed (high speed). The write buffer then completes the store to external memory at bus speed (typically slower than the core speed). During this time, the ARM9EJ-S processor can preform other tasks.

DCache and Write Buffer support write-back and write-through memory regions, controlled by C and B bits in each section and page descriptor within the MMU translation tables.

Write-though Operation

When a cache write hit occurs, the DCache line is updated. The updated data is then written to the write buffer which transfers it to external memory.

When a cache write miss occurs, a line, chosen by round robin or another algorithm, is stored in the write buffer which transfers it to external memory.

Write-back Operation

When a cache write hit occurs, the cache line or half line is marked as dirty, meaning that its contents are not up-to-date with those in the external memory.

When a cache write miss occurs, a line, chosen by round robin or another algorithm, is stored in the write buffer which transfers it to external memory.

11.7 Tightly-Coupled Memory Interface

11.7.1 TCM Description

The ARM926EJ-S processor features a Tightly-Coupled Memory (TCM) interface, which enables separate instruction and data TCMs (ITCM and DTCM) to be directly reached by the processor. TCMs are used to store real-time and performance critical code, they also provide a DMA support mechanism. Unlike AHB accesses to external memories, accesses to TCMs are fast and deterministic and do not incur bus penalties.

The user has the possibility to independently configure each TCM size with values within the following ranges, [0KB, 64 KB] for ITCM size and [0KB, 64 KB] for DTCM size.

TCMs can be configured by two means: the Bus Matrix TCM register and TCM region register (register 9) in CP15 and both steps should be performed. The Bus Matrix TCM register sets TCM size whereas TCM region register (register 9) in CP15 maps TCMs and enables them.

The data side of the ARM9EJ-S core is able to access the ITCM. This is necessary to enable code to be loaded into the ITCM, for SWI and emulated instruction handlers, and for accesses to PC-relative literal pools.



11.7.2 Enabling and Disabling TCMs

Prior to any enabling step, the user should configure the TCM sizes in the Bus Matrix TCM register. Then enabling TCMs is performed by using TCM region register (register 9) in CP15. The user should use the same sizes as those put in the Bus Matrix TCM register. For further details and programming tips, please refer to chapter 2.3 in ARM926EJ-S TRM, ref. DDI0222B.

11.7.3 TCM Mapping

The TCMs can be located anywhere in the memory map, with a single region available for ITCM and a separate region available for DTCM. The TCMs are physically addressed and can be placed anywhere in physical address space. However, the base address of a TCM must be aligned to its size, and the DTCM and ITCM regions must not overlap. TCM mapping is performed by using TCM region register (register 9) in CP15. The user should input the right mapping address for TCMs.

11.8 Bus Interface Unit

The ARM926EJ-S features a Bus Interface Unit (BIU) that arbitrates and schedules AHB requests. The BIU implements a multi-layer AHB, based on the AHB-Lite protocol, that enables parallel access paths between multiple AHB masters and slaves in a system. This is achieved by using a more complex interconnection matrix and gives the benefit of increased overall bus bandwidth, and a more flexible system architecture.

The multi-master bus architecture has a number of benefits:

- It allows the development of multi-master systems with an increased bus bandwidth and a flexible architecture.
- Each AHB layer becomes simple because it only has one master, so no arbitration or master-to-slave muxing is required. AHB layers, implementing AHB-Lite protocol, do not have to support request and grant, nor do they have to support retry and split transactions.
- The arbitration becomes effective when more than one master wants to access the same slave simultaneously.

11.8.1 Supported Transfers

The ARM926EJ-S processor performs all AHB accesses as single word, bursts of four words, or bursts of eight words. Any ARM9EJ-S core request that is not 1, 4, 8 words in size is split into packets of these sizes. Note that the Atmel bus is AHB-Lite protocol compliant, hence it does not support split and retry requests.

Table 8 gives an overview of the supported transfers and different kinds of transactions they are used for.

Table 11-7. Supported Transfers

HBurst[2:0]	Description	
SINGLE	Single transfer	Single transfer of word, half word, or byte: • data write (NCNB, NCB, WT, or WB that has missed in DCache) • data read (NCNB or NCB) • NC instruction fetch (prefetched and non-prefetched) • page table walk read
INCR4	Four-word incrementing burst	Half-line cache write-back, Instruction prefetch, if enabled. Four-word burst NCNB, NCB, WT, or WB write.
INCR8	Eight-word incrementing burst	Full-line cache write-back, eight-word burst NCNB, NCB, WT, or WB write.
WRAP8	Eight-word wrapping burst	Cache linefill

11.8.2 Thumb Instruction Fetches

All instructions fetches, regardless of the state of ARM9EJ-S core, are made as 32-bit accesses on the AHB. If the ARM9EJ-S is in Thumb state, then two instructions can be fetched at a time.

11.8.3 Address Alignment

The ARM926EJ-S BIU performs address alignment checking and aligns AHB addresses to the necessary boundary. 16-bit accesses are aligned to halfword boundaries, and 32-bit accesses are aligned to word boundaries.

12. AT91SAM9261 Debug and Test

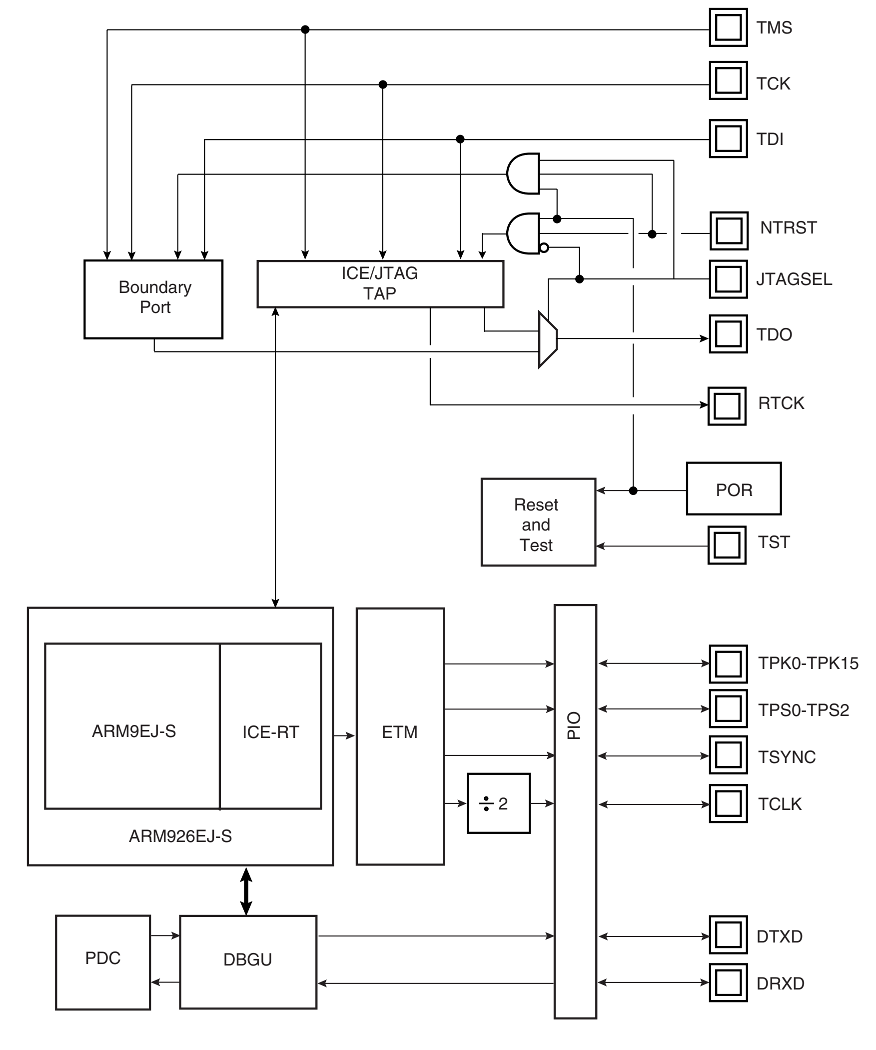
12.1 Overview

The AT91SAM9261 features a number of complementary debug and test capabilities. A common JTAG/ICE (In-Circuit Emulator) port is used for standard debugging functions, such as downloading code and single-stepping through programs. An ETM (Embedded Trace Macrocell) provides more sophisticated debug features such as address and data comparators, half-rate clock mode, counters, sequencer and FIFO. The Debug Unit provides a two-pin UART that can be used to upload an application into internal SRAM. It manages the interrupt handling of the internal COMMTX and COMMRX signals that trace the activity of the Debug Communication Channel.

A set of dedicated debug and test input/output pins gives direct access to these capabilities from a PC-based test environment.

12.2 Block Diagram

Figure 12-1. Debug and Test Block Diagram



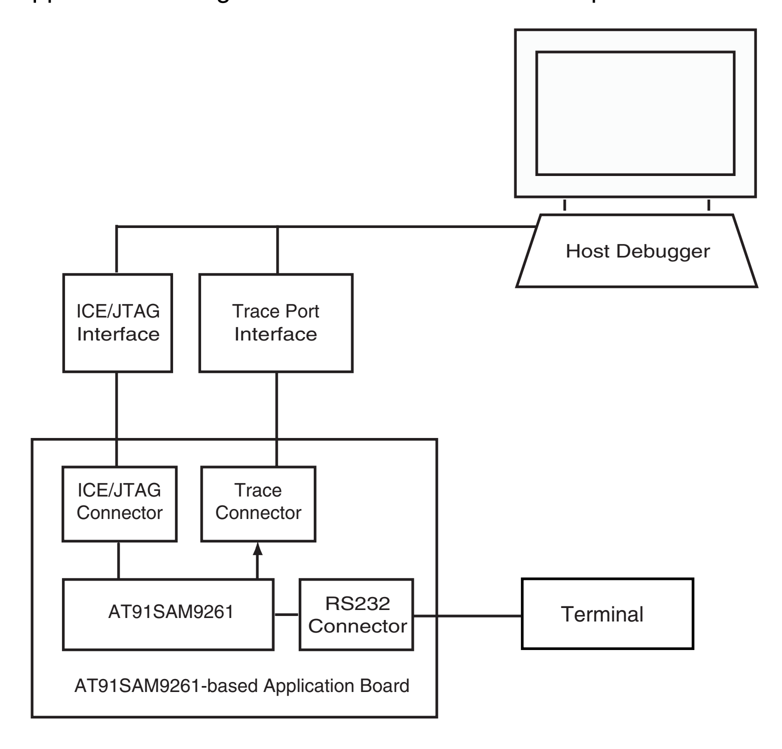
TAP: Test Access Port

12.3 Application Examples

12.3.1 Debug Environment

Figure 12-2 on page 59 shows a complete debug environment example. The ICE/JTAG interface is used for standard debugging functions, such as downloading code and single-stepping through the program. The Trace Port interface is used for tracing information. A software debugger running on a personal computer provides the user interface for configuring a Trace Port interface utilizing the ICE/JTAG interface.

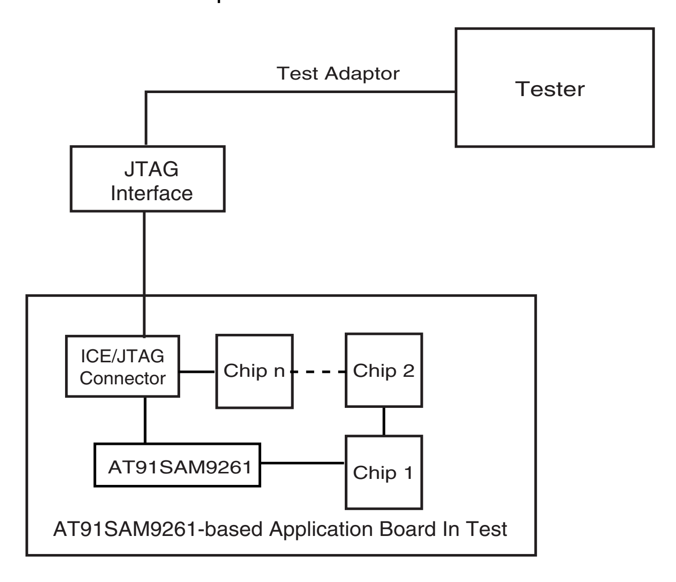
Figure 12-2. Application Debug and Trace Environment Example



12.3.2 Test Environment

Figure 12-3 on page 60 shows a test environment example. Test vectors are sent and interpreted by the tester. In this example, the "board in test" is designed using a number of JTAG-compliant devices. These devices can be connected to form a single scan chain.

Figure 12-3. Application Test Environment Example



12.4 Debug and Test Pin Description

Table 12-1. Debug and Test Pin List

Pin Name	Function	Type	Active Level
Reset/Test			
NRST	Microcontroller Reset	Input/Output	Low
TST	Test Mode Select	Input	High
ICE and JTAG			
TCK	Test Clock	Input	
TDI	Test Data In	Input	
TDO	Test Data Out	Output	
TMS	Test Mode Select	Input	
NTRST	Test Reset Signal	Input	Low
RTCK	Returned Test Clock	Output	
JTAGSEL	JTAG Selection	Input	
ETM			
TSYNC	Trace Synchronization Signal	Output	
TCLK	Trace Clock	Output	
TPS0 - TPS2	Trace ARM Pipeline Status	Output	
TPK0 - TPK15	Trace Packet Port	Output	

Table 12-1. Debug and Test Pin List

Debug Unit			
DRXD	Debug Receive Data	Input	
DTXD	Debug Transmit Data	Output	

12.5 Functional Description

12.5.1 Test Pin

One dedicated pin, TST, is used to define the device operating mode. The user must make sure that this pin is tied at low level to ensure normal operating conditions. Other values associated with this pin are reserved for manufacturing test.

12.5.2 Embedded In-circuit Emulator

The ARM9EJ-S Embedded In-Circuit Emulator-RT is supported via the ICE/JTAG port. It is connected to a host computer via an ICE interface. Debug support is implemented using an ARM9EJ-S core embedded within the ARM926EJ-S. The internal state of the ARM926EJ-S is examined through an ICE/JTAG port which allows instructions to be serially inserted into the pipeline of the core without using the external data bus. Therefore, when in debug state, a store-multiple (STM) can be inserted into the instruction pipeline. This exports the contents of the ARM9EJ-S registers. This data can be serially shifted out without affecting the rest of the system.

There are two scan chains inside the ARM9EJ-S processor which support testing, debugging, and programming of the Embedded ICE-RT. The scan chains are controlled by the ICE/JTAG port.

Embedded ICE mode is selected when JTAGSEL is low. It is not possible to switch directly between ICE and JTAG operations. A chip reset must be performed after JTAGSEL is changed.

For further details on the Embedded In-Circuit-Emulator-RT, see the ARM document: ARM9EJ-S Technical Reference Manual (DDI 0222A).

12.5.3 Debug Unit

The Debug Unit provides a two-pin (DXRD and TXRD) USART that can be used for several debug and trace purposes and offers an ideal means for in-situ programming solutions and debug monitor communication. Moreover, the association with two Peripheral DMA Controller channels permits packet handling of these tasks with processor time reduced to a minimum.

The Debug Unit also manages the interrupt handling of the COMMTX and COMMRX signals that come from the ICE and that trace the activity of the Debug Communication Channel.The Debug Unit allows blockage of access to the system through the ICE interface.

A specific register, the Debug Unit Chip ID Register, gives information about the product version and its internal configuration.

The AT91SAM9261 Debug Unit Chip ID value is 0x0197 03A0 on 32-bit width.

12.5.4 Embedded Trace Macrocell

The AT91SAM9261 features an Embedded Trace Macrocell (ETM), which is closely connected to the ARM926EJ-S Processor. The Embedded Trace is a standard Medium+ level implementation and contains the following resources:



- Four pairs of address comparators
- Two data comparators
- Eight memory map decoder inputs
- Two 16-bit counters
- One 3-stage sequencer
- Four external inputs
- One external output
- One 45-byte FIFO

The Embedded Trace Macrocell of the AT91SAM9261 works in half-rate clock mode and thus integrates a clock divider. Thus the maximum frequency of all the trace port signals does not exceed one half of the ARM926EJ-S clock speed.

The Embedded Trace Macrocell input and output resources are not used in the AT91SAM9261.

The Embedded Trace is a real-time trace module with the capability of tracing the ARM9EJ-S instructions and data.

For further details on Embedded Trace Macrocell, see the ARM documents:

- ETM9 (Rev2p2) Technical Reference Manual (DDI 0157F)
- Embedded Trace Macrocell Specification (IHI 0014J)

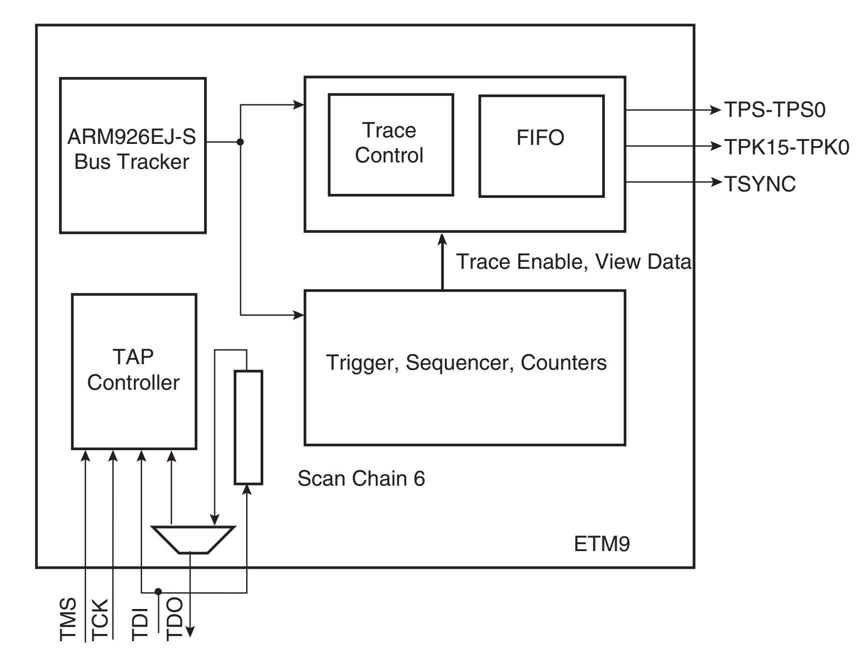
12.5.4.1 Trace Port

The Trace Port is made up of the following pins:

- TSYNC - the synchronization signal (Indicates the start of a branch sequence on the trace packet port.)
- TCLK - the Trace Port clock, half-rate of the ARM926EJ-S processor clock.
- TPS0 to TPS2 - indicate the processor state at each trace clock edge.
- TPK0 to TPK15 - the Trace Packet data value.

The trace packet information (address, data) is associated with the processor state indicated by TPS. Some processor states have no additional data associated with the Trace Packet Port (i.e., failed condition code of an instruction). The packet is 8 bits wide, and up to two packets can be output per cycle.

Figure 12-4. ETM9 Block



12.5.4.2 Implementation Details

This section gives an overview of the Embedded Trace resources.

Three-state Sequencer

The sequencer has three possible next states (one dedicated to itself and two others) and can change on every clock cycle. The state transition is controlled with internal events. If the user needs multiple-stage trigger schemes, the trigger event is based on a sequencer state.

Address Comparator

In single mode, address comparators compare either the instruction address or the data address against the user-programmed address.

In range mode, the address comparators are arranged in pairs to form a virtual address range resource.

Details of the address comparator programming are:

- The first comparator is programmed with the range start address.
- The second comparator is programmed with the range end address.
- The resource matches if the address is within the following range:
  – (address > = range start address) AND (address < range end address)
- Unpredictable behavior occurs if the two address comparators are not configured in the same way.

Data Comparator

Each full address comparator is associated with a specific data comparator. A data comparator is used to observe the data bus only when load and store operations occur.

A data comparator has both a value register and a mask register, therefore it is possible to compare only certain bits of a preprogrammed value against the data bus.

Memory Decoder Inputs

The eight memory map decoder inputs are connected to custom address decoders. The address decoders divide the memory into regions of on-chip SRAM, on-chip ROM, and peripherals. The address decoders also optimize the ETM9 trace trigger.

Table 12-2. ETM Memory Map Inputs Layout

Product Resource	Area	Access Type	Start Address	End Address
SRAM	Internal	Data	0x0000 0000	0x002F FFFF
SRAM	Internal	Fetch	0x0000 0000	0x002F FFFF
ROM	Internal	Data	0x0040 0000	0x004F FFFF
ROM	Internal	Fetch	0x0040 0000	0x004F FFFF
External Bus Interface	External	Data	0x1000 0000	0x8FFF FFFF
External Bus Interface	External	Fetch	0x1000 0000	0x8FFF FFFF
User Peripherals	Internal	Data	0xF000 0000	0xFFFF BFFF
System Peripherals	Internal	Data	0xFFFF C000	0xFFFF FFFF

FIFO

A 45-byte FIFO is used to store data tracing. The FIFO is used to separate the pipeline status from the trace packet, thus the FIFO can be used to buffer trace packets.

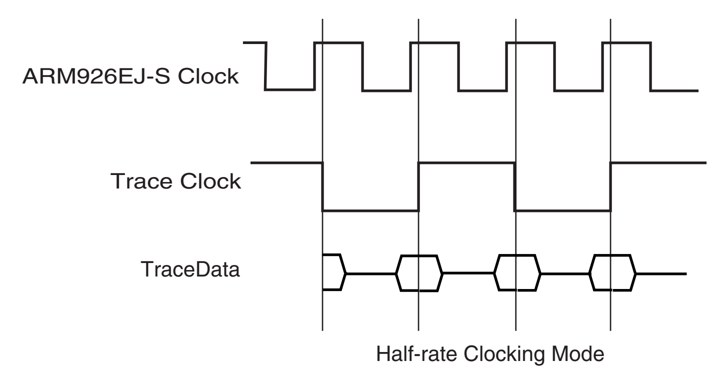
A FIFO overflow is detected by the embedded trace macrocell when the FIFO is full or when the FIFO has fewer bytes than the user-programmed number.

Half-rate Clocking Mode

The ETM9 is implemented in half-rate mode that allows both rising and falling edge data tracing of the trace clock.

The half-rate mode is implemented to maintain the signal clock integrity of high-speed systems (up to 100 Mhz).

Figure 12-5. Half-rate Clocking Mode



Care must be taken on the choice of the trace capture system as it needs to support half-rate clock functionality.

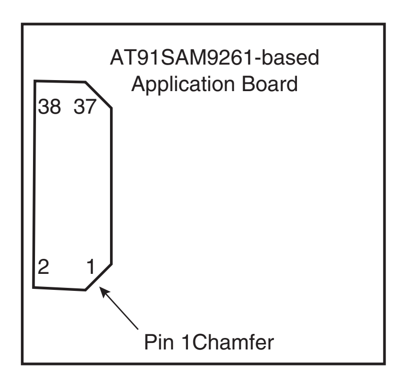
12.5.4.3 Application Board Restriction

The TCLK signal needs to be set with care, some timing parameters are required.

The specified target system connector is the AMP Mictor connector.

The connector must be oriented on the application board as shown in Figure 12-6. The view of the PCB is shown from above with the trace connector mounted near the edge of the board. This allows the Trace Port Analyzer to minimize the physical intrusiveness of the interconnected target.

Figure 12-6. AMP Mictor Connector Orientation



12.5.5 IEEE 1149.1 JTAG Boundary Scan

IEEE 1149.1 JTAG Boundary Scan allows pin-level access independent of the device packaging technology.

IEEE 1149.1 JTAG Boundary Scan is enabled when JTAGSEL is high. The SAMPLE, EXTEST and BYPASS functions are implemented. In ICE debug mode, the ARM processor responds with a non-JTAG chip ID that identifies the processor to the ICE system. This is not IEEE 1149.1 JTAG-compliant.

It is not possible to switch directly between JTAG and ICE operations. A chip reset must be performed after JTAGSEL is changed.

A Boundary-scan Descriptor Language (BSDL) file is provided to set up test.

12.5.5.1 JTAG Boundary-scan Register

The Boundary-scan Register (BSR) contains 484 bits that correspond to active pins and associated control signals.

Each AT91SAM9261 input/output pin corresponds to a 3-bit register in the BSR. The OUTPUT bit contains data that can be forced on the pad. The INPUT bit facilitates the observability of data applied to the pad. The CONTROL bit selects the direction of the pad.

Table 12-3. AT91SAM9261 JTAG Boundary Scan Register

Bit Number	Pin Name	Pin Type	Associated BSR Cells
483	PC23	IN/OUT	INPUT
482			OUTPUT
481			CONTROL
480		internal	



Table 12-3. AT91SAM9261 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
479	PC24	IN/OUT	INPUT
478			OUTPUT
477			CONTROL
476		internal	
475	PC25	IN/OUT	INPUT
474			OUTPUT
473			CONTROL
472		internal	
471	PC26	IN/OUT	INPUT
470			OUTPUT
469			CONTROL
468		internal	
467	PC27	IN/OUT	INPUT
466			OUTPUT
465			CONTROL
464		internal	
463	PC28	IN/OUT	INPUT
462			OUTPUT
461			CONTROL
460		internal	
459	PC29	IN/OUT	INPUT
458			OUTPUT
457			CONTROL
456		internal	
455	PC0	IN/OUT	INPUT
454			OUTPUT
453			CONTROL
452		internal	
451	PC1	IN/OUT	INPUT
450			OUTPUT
449			CONTROL
448		internal	
447	PC2	IN/OUT	INPUT
446			OUTPUT
445			CONTROL
444		internal	

Table 12-3. AT91SAM9261 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
443	PC3	IN/OUT	INPUT
442			OUTPUT
441			CONTROL
440		internal	
439	PC4	IN/OUT	INPUT
438			OUTPUT
437			CONTROL
436		internal	
435	PC5	IN/OUT	INPUT
434			OUTPUT
433			CONTROL
432		internal	
431	PC6	IN/OUT	INPUT
430			OUTPUT
429			CONTROL
428		internal	
427	PC7	IN/OUT	INPUT
426			OUTPUT
425			CONTROL
424		internal	
423	PC8	IN/OUT	INPUT
422			OUTPUT
421			CONTROL
420		internal	
419	PC9	IN/OUT	INPUT
418			OUTPUT
417			CONTROL
416		internal	
415	PC10	IN/OUT	INPUT
414			OUTPUT
413			CONTROL
412		internal	
411	PC11	IN/OUT	INPUT
410			OUTPUT
409			CONTROL
408		internal	

Table 12-3. AT91SAM9261 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
407	PC12	IN/OUT	INPUT
406			OUTPUT
405			CONTROL
404		internal	
403	PC13	IN/OUT	INPUT
402			OUTPUT
401			CONTROL
400		internal	
399	PC14	IN/OUT	INPUT
398			OUTPUT
397			CONTROL
396		internal	
395	PC15	IN/OUT	INPUT
394			OUTPUT
393			CONTROL
392		internal	
391	PA0	IN/OUT	INPUT
390			OUTPUT
389			CONTROL
388		internal	
387	PA1	IN/OUT	INPUT
386			OUTPUT
385			CONTROL
384		internal	
383	PA2	IN/OUT	INPUT
382			OUTPUT
381			CONTROL
380		internal	
379	PA3	IN/OUT	INPUT
378			OUTPUT
377			CONTROL
376		internal	
375	PA4	IN/OUT	INPUT
374			OUTPUT
373			CONTROL
372		internal	

Table 12-3. AT91SAM9261 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
371	PA5	IN/OUT	INPUT
370			OUTPUT
369			CONTROL
368		internal	
367	PA6	IN/OUT	INPUT
366			OUTPUT
365			CONTROL
364		internal	
363	PA7	IN/OUT	INPUT
362			OUTPUT
361			CONTROL
360		internal	
359	PA12	IN/OUT	INPUT
358			OUTPUT
357			CONTROL
356		internal	
355	PA8	IN/OUT	INPUT
354			OUTPUT
353			CONTROL
352		internal	
351	PA9	IN/OUT	INPUT
350			OUTPUT
349			CONTROL
348		internal	
347	PA10	IN/OUT	INPUT
346			OUTPUT
345			CONTROL
344		internal	
343	PA11	IN/OUT	INPUT
342			OUTPUT
341			CONTROL
340		internal	
339	PA13	IN/OUT	INPUT
338			OUTPUT
337			CONTROL
336		internal	



Table 12-3. AT91SAM9261 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
335	PA14	IN/OUT	INPUT
334			OUTPUT
333			CONTROL
332		internal	
331	PA15	IN/OUT	INPUT
330			OUTPUT
329			CONTROL
328		internal	
327	PA16	IN/OUT	INPUT
326			OUTPUT
325			CONTROL
324		internal	
323	PA17	IN/OUT	INPUT
322			OUTPUT
321			CONTROL
320		internal	
319	PA18	IN/OUT	INPUT
318			OUTPUT
317			CONTROL
316		internal	
315	PA19	IN/OUT	INPUT
314			OUTPUT
313			CONTROL
312		internal	
311	PA20	IN/OUT	INPUT
310			OUTPUT
309			CONTROL
308		internal	
307	PA21	IN/OUT	INPUT
306			OUTPUT
305			CONTROL
304		internal	
303	PA22	IN/OUT	INPUT
302			OUTPUT
301			CONTROL
300		internal	

Table 12-3. AT91SAM9261 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
299	PA23	IN/OUT	INPUT
298			OUTPUT
297			CONTROL
296		internal	
295	PA24	IN/OUT	INPUT
294			OUTPUT
293			CONTROL
292		internal	
291	PA25	IN/OUT	INPUT
290			OUTPUT
289			CONTROL
288		internal	
287	PA26	IN/OUT	INPUT
286			OUTPUT
285			CONTROL
284		internal	
283	PA27	IN/OUT	INPUT
282			OUTPUT
281			CONTROL
280		internal	
279	PA28	IN/OUT	INPUT
278			OUTPUT
277			CONTROL
276		internal	
275	PA29	IN/OUT	INPUT
274			OUTPUT
273			CONTROL
272		internal	
271	PA30	IN/OUT	INPUT
270			OUTPUT
269			CONTROL
268		internal	
267	PA31	IN/OUT	INPUT
266			OUTPUT
265			CONTROL
264		internal	

Table 12-3. AT91SAM9261 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
263	PB0	IN/OUT	INPUT
262			OUTPUT
261			CONTROL
260		internal	
259	PB1	IN/OUT	INPUT
258			OUTPUT
257			CONTROL
256		internal	
255	PB2	IN/OUT	INPUT
254			OUTPUT
253			CONTROL
252		internal	
251	PB3	IN/OUT	INPUT
250			OUTPUT
249			CONTROL
248		internal	
247	PB4	IN/OUT	INPUT
246			OUTPUT
245			CONTROL
244		internal	
243	PB5	IN/OUT	INPUT
242			OUTPUT
241			CONTROL
240		internal	
239	PB6	IN/OUT	INPUT
238			OUTPUT
237			CONTROL
236		internal	
235	PB7	IN/OUT	INPUT
234			OUTPUT
233			CONTROL
232		internal	
231	PB8	IN/OUT	INPUT
230			OUTPUT
229			CONTROL
228		internal	

Table 12-3. AT91SAM9261 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
227	PB9	IN/OUT	INPUT
226			OUTPUT
225			CONTROL
224		internal	
223	PB10	IN/OUT	INPUT
222			OUTPUT
221			CONTROL
220		internal	
219	PB11	IN/OUT	INPUT
218			OUTPUT
217			CONTROL
216		internal	
215	PB12	IN/OUT	INPUT
214			OUTPUT
213			CONTROL
212		internal	
211	PB13	IN/OUT	INPUT
210			OUTPUT
209			CONTROL
208		internal	
207	PB14	IN/OUT	INPUT
206			OUTPUT
205			CONTROL
204		internal	
203	PB15	IN/OUT	INPUT
202			OUTPUT
201			CONTROL
200		internal	
199	PB16	IN/OUT	INPUT
198			OUTPUT
197			CONTROL
196		internal	
195	PB17	IN/OUT	INPUT
194			OUTPUT
193			CONTROL
192		internal	

Table 12-3. AT91SAM9261 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
191	PB18	IN/OUT	INPUT
190			OUTPUT
189			CONTROL
188		internal	
187	PB19	IN/OUT	INPUT
186			OUTPUT
185			CONTROL
184		internal	
183	PB20	IN/OUT	INPUT
182			OUTPUT
181			CONTROL
180		internal	
179	PB21	IN/OUT	INPUT
178			OUTPUT
177			CONTROL
176		internal	
175	PB22	IN/OUT	INPUT
174			OUTPUT
173			CONTROL
172		internal	
171	PB23	IN/OUT	INPUT
170			OUTPUT
169			CONTROL
168		internal	
167	PB24	IN/OUT	INPUT
166			OUTPUT
165			CONTROL
164		internal	
163	PB25	IN/OUT	INPUT
162			OUTPUT
161			CONTROL
160		internal	
159	PB26	IN/OUT	INPUT
158			OUTPUT
157			CONTROL
156		internal	

Table 12-3. AT91SAM9261 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
155	PB27	IN/OUT	INPUT
154			OUTPUT
153			CONTROL
152		internal	
151	PB28	IN/OUT	INPUT
150			OUTPUT
149			CONTROL
148		internal	
147	PB29	IN/OUT	INPUT
146			OUTPUT
145			CONTROL
144		internal	
143	PB30	IN/OUT	INPUT
142			OUTPUT
141			CONTROL
140		internal	
139	PB31	IN/OUT	INPUT
138			OUTPUT
137			CONTROL
136		internal	
135	A0/NBS0	OUT	OUTPUT
134	A[7:0]		CONTROL
133		internal	
132	A1/NBS2/NWR2	OUT	OUTPUT
131	A2	OUT	OUTPUT
130	A3	OUT	OUTPUT
129	A4	OUT	OUTPUT
128	A5	OUT	OUTPUT
127	A6	OUT	OUTPUT
126	A7	OUT	OUTPUT
125	A8	OUT	OUTPUT
124	A[15:8]		CONTROL
123	A9	OUT	OUTPUT
122	A10	OUT	OUTPUT
121	SDA10	IN/OUT	OUTPUT

Table 12-3. AT91SAM9261 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
120	SDA10/SDRAMCKE RAS/CAS/SDWE		CONTROL
119	A11	OUT	OUTPUT
118	A12	OUT	OUTPUT
117	A13	OUT	OUTPUT
116	A14	OUT	OUTPUT
115	A15	OUT	OUTPUT
114	A16	OUT	OUTPUT
113	A[22:16]		CONTROL
112	A17	OUT	OUTPUT
111	A18	OUT	OUTPUT
110	A19	OUT	OUTPUT
109	A20	OUT	OUTPUT
108	A21	OUT	OUTPUT
107	A22	OUT	OUTPUT
106	NCS0	OUT	OUTPUT
105	NCS0/NCS1/NCS2/NCS3 NRD/NWR0/NWR1/NWR3		CONTROL
104	NCS1/NDCS	OUT	OUTPUT
103	NCS2	OUT	OUTPUT
102	NCS3/NANDCS	OUT	OUTPUT
101	NRD/CFOE	OUT	OUTPUT
100	NWR0/NWE/CFWE	IN/OUT	INPUT
99			OUTPUT
98	NWR1/NBS1/CFIOR	IN/OUT	INPUT
97			OUTPUT
96		internal	
95	NWR3/NBS3/CFIOW	OUT	OUTPUT
94	SDRAMCKE	OUT	OUTPUT
93	SDRAMCLK	IN/OUT	INPUT
92			OUTPUT
91			CONTROL
90	RAS	OUT	OUTPUT
89	CAS	OUT	OUTPUT
88	SDWE	OUT	OUTPUT

Table 12-3. AT91SAM9261 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
87	D0	IN/OUT	INPUT
86			OUTPUT
85			CONTROL
84		internal	
83	D1	IN/OUT	INPUT
82			OUTPUT
81			CONTROL
80	D2	IN/OUT	INPUT
79			OUTPUT
78			CONTROL
77	D3	IN/OUT	INPUT
76			OUTPUT
75			CONTROL
74	D4	IN/OUT	INPUT
73			OUTPUT
72			CONTROL
71		internal	
70	D5	IN/OUT	INPUT
69			OUTPUT
68			CONTROL
67	D6	IN/OUT	INPUT
66			OUTPUT
65			CONTROL
64	D7	IN/OUT	INPUT
63			OUTPUT
62			CONTROL
61	D8	IN/OUT	INPUT
60			OUTPUT
59			CONTROL
58		internal	
57	D9	IN/OUT	INPUT
56			OUTPUT
55			CONTROL
54	D10	IN/OUT	INPUT
53			OUTPUT
52			CONTROL

Table 12-3. AT91SAM9261 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
51	D11	IN/OUT	INPUT
50			OUTPUT
49			CONTROL
48	D12	IN/OUT	INPUT
47			OUTPUT
46			CONTROL
45		internal	
44	D13	IN/OUT	INPUT
43			OUTPUT
42			CONTROL
41	D14	IN/OUT	INPUT
40			OUTPUT
39			CONTROL
38	D15	IN/OUT	INPUT
37			OUTPUT
36			CONTROL
35	PC16	IN/OUT	INPUT
34			OUTPUT
33			CONTROL
32		internal	
31	PC17	IN/OUT	INPUT
30			OUTPUT
29			CONTROL
28		internal	
27	PC18	IN/OUT	INPUT
26			OUTPUT
25			CONTROL
24		internal	
23	PC19	IN/OUT	INPUT
22			OUTPUT
21			CONTROL
20		internal	
19	PC30	IN/OUT	INPUT
18			OUTPUT
17			CONTROL
16		internal	

Table 12-3. AT91SAM9261 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
15	PC31	IN/OUT	INPUT
14			OUTPUT
13			CONTROL
12		internal	
11	PC20	IN/OUT	INPUT
10			OUTPUT
09			CONTROL
08		internal	
07	PC21	IN/OUT	INPUT
06			OUTPUT
05			CONTROL
04		internal	
03	PC22	IN/OUT	INPUT
02			OUTPUT
01			CONTROL
00		internal	

12.5.6 ID Code Register

Access: Read-only

31	30	29	28	27	26	25	24
VERSION				PART NUMBER			

23	22	21	20	19	18	17	16
PART NUMBER							

15	14	13	12	11	10	9	8
PART NUMBER				MANUFACTURER IDENTITY			

7	6	5	4	3	2	1	0
MANUFACTURER IDENTITY							1

- **VERSION[31:28]: Product Version Number**

Set to 0x0.

- **PART NUMBER[27:12]: Product Part Number**

Product part Number is 0x5B08

- **MANUFACTURER IDENTITY[11:1]**

Set to 0x01F.

Bit[0] Required by IEEE Std. 1149.1.

Set to 0x1.

JTAG ID Code value is 0x05B0_803F.

13. AT91SAM Boot Program

13.1 Description

The Boot Program integrates different programs permitting download and/or upload into the different memories of the product.

First, it initializes the Debug Unit serial port (DBGU) and the USB Device Port.

Then the DataFlash® Boot program is executed. It looks for a sequence of eight valid ARM exception vectors in a DataFlash connected to the SPI. All these vectors must be B-branch or LDR load register instructions except for the sixth vector. This vector is used to store the size of the image to download.

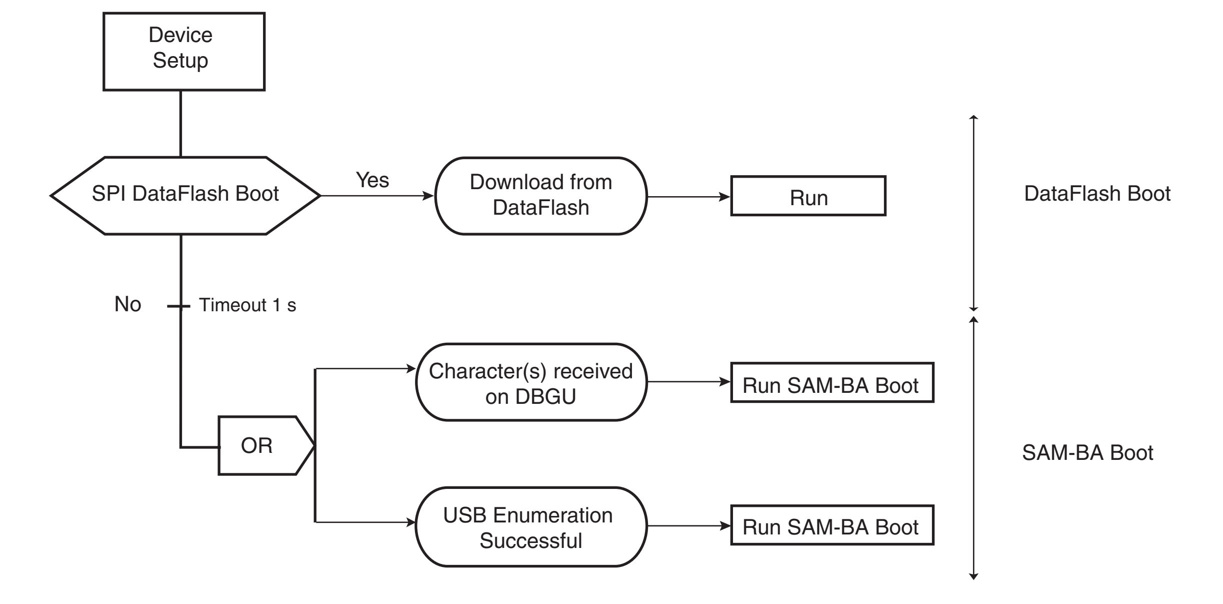
If a valid sequence is found, code is downloaded into the internal SRAM. This is followed by a remap and a jump to the first address of the SRAM.

If no valid ARM vector sequence is found, SAM-BA™ Boot is then executed. It waits for transactions either on the USB device, or on the DBGU serial port.

13.2 Flow Diagram

The Boot Program implements the algorithm in Figure 13-1.

Figure 13-1. Boot Program Algorithm Flow Diagram



13.3 Device Initialization

Initialization follows the steps described below:

1. Stack setup for ARM supervisor mode
2. Main oscillator frequency detection
3. C variable initialization
4. PLL setup: PLLB is initialized to generate a 48 MHz clock necessary to use the USB Device. A register located in the Power Management Controller (PMC) determines the frequency of the main oscillator and thus the correct factor for the PLLB.

Table 13-1 defines the crystals supported by the DataFlash Boot Program.

Table 13-1. Crystals Supported by Software Auto-detection (MHz)

3.0	3.2768	3.6864	3.84	4.0
4.433619	4.608	4.9152	5.0	5.24288
6.0	6.144	6.4	6.5536	7.159090
7.3728	7.864320	8.0	9.8304	10.0
11.05920	12.0	12.288	13.56	14.31818
14.7456	16.0	17.734470	18.432	20.0

5. Initialization of the DBGU serial port (115200 bauds, 8, N, 1)
6. Disable the Watchdog and enable the user reset
7. Initialization of the USB Device Port
8. Jump to DataFlash Boot sequence

If DataFlash Boot fails:

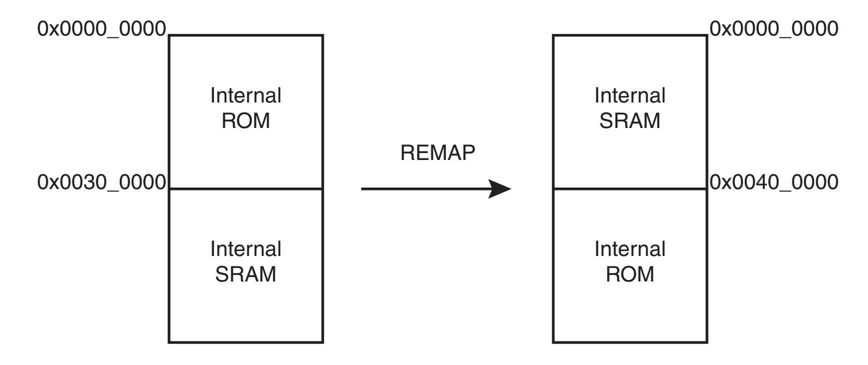
9. Activation of the Instruction Cache
10. Jump to SAM-BA Boot sequence

13.4 DataFlash Boot

The DataFlash Boot program searches for a valid application in the SPI DataFlash memory. If a valid application is found, this application is loaded into internal SRAM and executed by branching at address 0x0000_0000 after remap. This application may be the application code or a second-level bootloader.

All the calls to functions are PC relative and do not use absolute addresses.

Figure 13-2. Remap Action after Download Completion



After reset, the code in internal ROM is mapped at both addresses 0x0000_0000 and 0x0040_0000:

```
400000   ea000006   B      0x20     00   ea000006   B      0x20
400004   eaffffe    B      0x04     04   eaffffe    B      0x04
400008   ea00002f   B      _main    08   ea00002f   B      _main
40000c   eaffffe    B      0x0c     0c   eaffffe    B      0x0c
400010   eaffffe    B      0x10     10   eaffffe    B      0x10
400014   eaffffe    B      0x14     14   eaffffe    B      0x14
400018   eaffffe    B      0x18     18   eaffffe    B      0x18
40001c   eaffffe    B      0x1c     1c   eaffffe    B      0x1c
```

13.4.1 Valid Image Detection

The DataFlash Boot software looks for a valid application by analyzing the first 32 bytes corresponding to the ARM exception vectors. These bytes must implement ARM instructions for either branch or load PC with PC relative addressing.

The sixth vector, at offset 0x14, contains the size of the image to download. The user must replace this vector with his own vector (see "Structure of ARM Vector 6" on page 83).

Figure 13-3. LDR Opcode

31			28	27			24	23			20	19 … 16	15 … 12	11 … 0
1	1	1	0	1	1	I	P	U	1	W	0	Rn	Rd	

Figure 13-4. B Opcode

31			28	27			24	23 … 0
1	1	1	0	1	0	1	0	Offset (24 bits)

Unconditional instruction: 0xE for bits 31 to 28

Load PC with PC relative addressing instruction:

– Rn = Rd = PC = 0xF

– I==1

– P==1

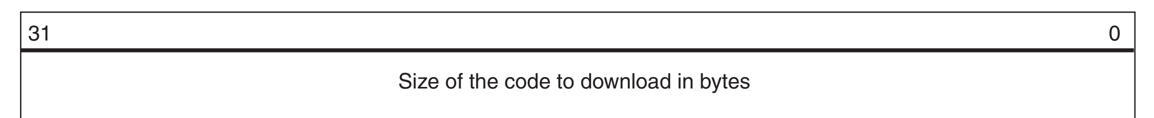
– U offset added (U==1) or subtracted (U==0)

– W==1

13.4.2 Structure of ARM Vector 6

The ARM exception vector 6 is used to store information needed by the DataFlash boot program.

Figure 13-5. Structure of the ARM vector 6



13.4.2.1 Example

An example of valid vectors follows:

```
00    ea000006   B      0x20
04    eafffffe   B      0x04
08    ea00002f   B      _main
0c    eafffffe   B      0x0c
10    eafffffe   B      0x10
14    00001234                  <- Code size = 4660 bytes
18    eafffffe   B      0x18
1c    eafffffe   B      0x1c
```

The size of the image to load into SRAM is contained in the location of the sixth ARM vector. Thus the user must replace this vector by the correct vector for his application.

13.4.3 DataFlash Boot Sequence

The DataFlash boot program performs device initialization followed by the download procedure.

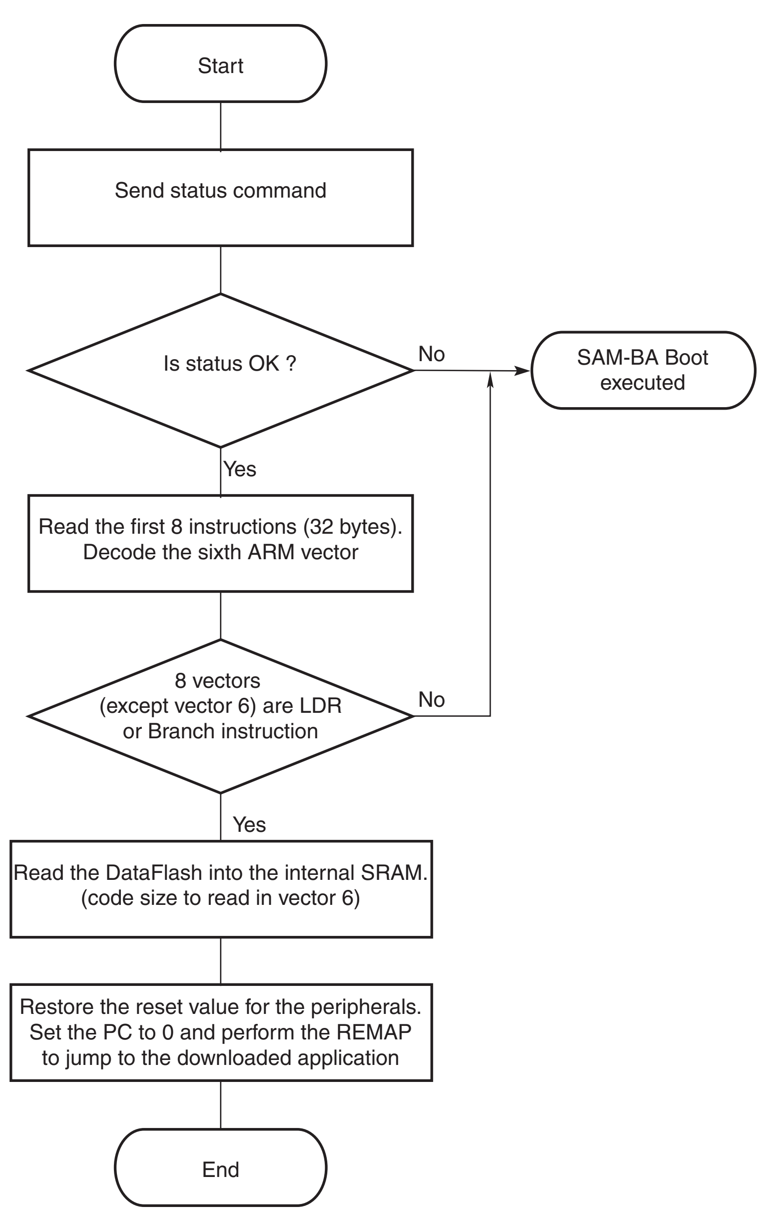
The DataFlash boot program supports all Atmel DataFlash devices. Table 13-2 summarizes the parameters to include in the ARM vector 6 for all devices.

Table 13-2. DataFlash Device

Device	Density	Page Size (bytes)	Number of Pages
AT45DB011B	1 Mbit	264	512
AT45DB021B	2 Mbits	264	1024
AT45DB041B	4 Mbits	264	2048
AT45DB081B	8 Mbits	264	4096
AT45DB161B	16 Mbits	528	4096
AT45DB321B	32 Mbits	528	8192
AT45DB642	64 Mbits	1056	8192

The DataFlash has a Status Register that determines all the parameters required to access the device. The DataFlash boot is configured to be compatible with the future design of the DataFlash.

Figure 13-6. Serial DataFlash Download



13.5 SAM-BA Boot

If no valid DataFlash device has been found during the DataFlash boot sequence, the SAM-BA boot program is performed.

The SAM-BA boot principle is to:

– Wait for USB Device enumeration

– In parallel, wait for character(s) received on the DBGU



– Once the communication interface is identified, the application runs in an infinite loop waiting for different commands as in Table 13-3.

Table 13-3. Commands Available through the SAM-BA Boot

Command	Action	Argument(s)	Example
O	write a byte	Address, Value#	**O**200001,CA#
o	read a byte	Address,#	**o**200001,#
H	write a half word	Address, Value#	**H**200002,CAFE#
h	read a half word	Address,#	**h**200002,#
W	write a word	Address, Value#	**W**200000,CAFEDECA#
w	read a word	Address,#	**w**200000,#
S	send a file	Address,#	**S**200000,#
R	receive a file	Address, NbOfBytes#	**R**200000,1234#
G	go	Address#	**G**200200#
V	display version	No argument	**V**#

- Write commands: Write a byte (**O**), a halfword (**H**) or a word (**W**) to the target.
  – *Address*: Address in hexadecimal.
  – *Value*: Byte, halfword or word to write in hexadecimal.
  – *Output*: '>'.
- Read commands: Read a byte (**o**), a halfword (**h**) or a word (**w**) from the target.
  – *Address*: Address in hexadecimal
  – *Output*: The byte, halfword or word read in hexadecimal following by '>'
- Send a file (**S**): Send a file to a specified address
  – *Address*: Address in hexadecimal
  – *Output*: '>'.

Note: There is a time-out on this command which is reached when the prompt '>' appears before the end of the command execution.

- Receive a file (**R**): Receive data into a file from a specified address
  – *Address*: Address in hexadecimal
  – *NbOfBytes*: Number of bytes in hexadecimal to receive
  – *Output*: '>'
- Go (**G**): Jump to a specified address and execute the code
  – *Address*: Address to jump in hexadecimal
  – *Output*: '>'
- Get Version (**V**): Return the SAM-BA boot version
  – *Output*: '>'

13.5.1 DBGU Serial Port

Communication is performed through the DBGU serial port initialized to 115200 Baud, 8, n, 1.

The Send and Receive File commands use the Xmodem protocol to communicate. Any terminal performing this protocol can be used to send the application file to the target. The size of

the binary file to send depends on the SRAM size embedded in the product. In all cases, the size of the binary file must be lower than the SRAM size because the Xmodem protocol requires some SRAM memory to work.

13.5.2 Xmodem Protocol

The Xmodem protocol supported is the 128-byte length block. This protocol uses a two-character CRC-16 to guarantee detection of a maximum bit error.

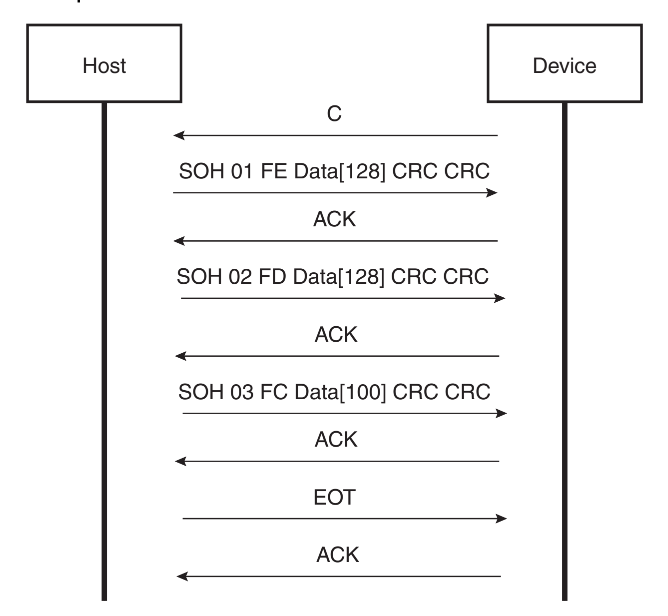
Xmodem protocol with CRC is accurate provided both sender and receiver report successful transmission. Each block of the transfer looks like:

<SOH><blk #><255-blk #><--128 data bytes--><checksum> in which:

- <SOH> = 01 hex
- <blk #> = binary number, starts at 01, increments by 1, and wraps 0FFH to 00H (not to 01)
- <255-blk #> = 1's complement of the blk#.
- <checksum> = 2 bytes CRC16

Figure 13-7 shows a transmission using this protocol.

Figure 13-7. Xmodem Transfer Example



13.5.3 USB Device Port

A 48 MHz USB clock is necessary to use the USB Device port. It has been programmed earlier in the device initialization procedure with PLLB configuration.

The device uses the USB communication device class (CDC) drivers to take advantage of the installed PC RS-232 software to talk over the USB. The CDC class is implemented in all releases of Windows®, from Windows 98SE to Windows XP. The CDC document, available at www.usb.org, describes a way to implement devices such as ISDN modems and virtual COM ports.

The Vendor ID is Atmel's vendor ID 0x03EB. The product ID is 0x6124. These references are used by the host operating system to mount the correct driver. On Windows systems, the INF files contain the correspondence between vendor ID and product ID.

Atmel provides an INF example to see the device as a new serial port and also provides another custom driver used by the SAM-BA application: atm6124.sys. Refer to the document "USB Basic Application", literature number 6123, for more details.

13.5.3.1 *Enumeration Process*

The USB protocol is a master/slave protocol. This is the host that starts the enumeration sending requests to the device through the control endpoint. The device handles standard requests as defined in the USB Specification.

Table 13-4. Handled Standard Requests

Request	Definition
GET_DESCRIPTOR	Returns the current device configuration value.
SET_ADDRESS	Sets the device address for all future device access.
SET_CONFIGURATION	Sets the device configuration.
GET_CONFIGURATION	Returns the current device configuration value.
GET_STATUS	Returns status for the specified recipient.
SET_FEATURE	Used to set or enable a specific feature.
CLEAR_FEATURE	Used to clear or disable a specific feature.

The device also handles some class requests defined in the CDC class.

Table 13-5. Handled Class Requests

Request	Definition
SET_LINE_CODING	Configures DTE rate, stop bits, parity and number of character bits.
GET_LINE_CODING	Requests current DTE rate, stop bits, parity and number of character bits.
SET_CONTROL_LINE_STATE	RS-232 signal used to tell the DCE device the DTE device is now present.

Unhandled requests are STALLed.

13.5.3.2 *Communication Endpoints*

There are two communication endpoints and endpoint 0 is used for the enumeration process. Endpoint 1 is a 64-byte Bulk OUT endpoint and endpoint 2 is a 64-byte Bulk IN endpoint. SAM-BA Boot commands are sent by the host through the endpoint 1. If required, the message is split by the host into several data payloads by the host driver.

If the command requires a response, the host can send IN transactions to pick up the response.

13.6 Hardware and Software Constraints

- The DataFlash downloaded code size must be inferior to 156K bytes.
- The code is always downloaded from the device address 0x0000_0000 to the address 0x0000_0000 of the internal SRAM (after remap).
- The downloaded code must be position-independent or linked at address 0x0000_0000.
- SAM-BA boot uses a block of internal SRAM for variables and stacks. The remaining available size for the user code is 156K bytes. This user area extends from address 0x0030_0000 to address 0x0032_6FFF.
- The DataFlash must be connected to NPCS0 of the SPI.

The SPI drivers use several PIOs in alternate functions to communicate with devices. Care must be taken when these PIOs are used by the application. The devices connected could be unintentionally driven at boot time, and electrical conflicts between SPI output pins and the connected devices may appear.

To assure correct functionality, it is recommended to plug in critical devices to other pins.

Table 13-6 contains a list of pins that are driven during the boot program execution. These pins are driven during the boot sequence for a period of less than 1 s if no correct boot program is found.

For the DataFlash driven by the SPCK signal at 8 MHz, the time to download 156K bytes is reduced to 200 ms.

Before performing the jump to the application in internal SRAM, all the PIOs and peripherals used in the boot program are set to their reset state.

Table 13-6. Pins Driven during Boot Program Execution

Peripheral	Pin	PIO Line
SPI0	MOSI	PA1
SPI0	MISO	PA0
SPI0	SPCK	PA2
SPI0	NPCS0	PA3
DBGU	DRXD	PA9
DBGU	DTXD	PA10

14. Reset Controller (RSTC)

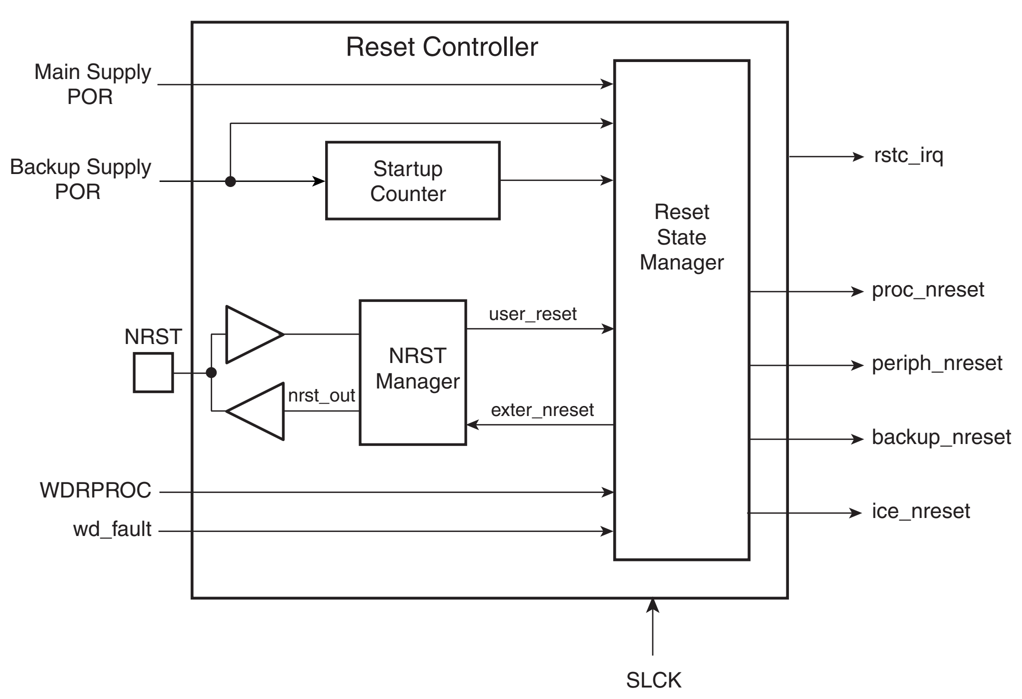
14.1 Description

The Reset Controller (RSTC), based on power-on reset cells, handles all the resets of the system without any external components. It reports which reset occurred last.

The Reset Controller also drives independently or simultaneously the external reset and the peripheral and processor resets.

14.2 Block Diagram

Figure 14-1. Reset Controller Block Diagram



14.3 Functional Description

14.3.1 Reset Controller Overview

The Reset Controller is made up of an NRST Manager, a Startup Counter and a Reset State Manager. It runs at Slow Clock and generates the following reset signals:

- proc_nreset: Processor reset line. It also resets the Watchdog Timer.
- backup_nreset: Affects all the peripherals powered by VDDBU.
- periph_nreset: Affects the whole set of embedded peripherals.
- nrst_out: Drives the NRST pin.

These reset signals are asserted by the Reset Controller, either on external events or on software action. The Reset State Manager controls the generation of reset signals and provides a signal to the NRST Manager when an assertion of the NRST pin is required.

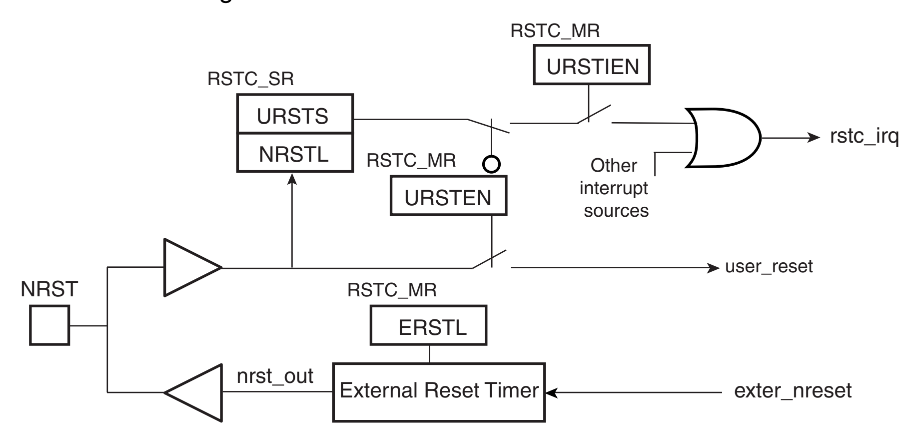
The NRST Manager shapes the NRST assertion during a programmable time, thus controlling external device resets.

The Reset Controller Mode Register (RSTC_MR), allowing the configuration of the Reset Controller, is powered with VDDBU, so that its configuration is saved as long as VDDBU is on.

14.3.2 NRST Manager

The NRST Manager samples the NRST input pin and drives this pin low when required by the Reset State Manager. Figure 14-2 shows the block diagram of the NRST Manager.

Figure 14-2. NRST Manager



14.3.2.1 NRST Signal or Interrupt

The NRST Manager samples the NRST pin at Slow Clock speed. When the line is detected low, a User Reset is reported to the Reset State Manager.

However, the NRST Manager can be programmed to not trigger a reset when an assertion of NRST occurs. Writing the bit URSTEN at 0 in RSTC_MR disables the User Reset trigger.

The level of the pin NRST can be read at any time in the bit NRSTL (NRST level) in RSTC_SR. As soon as the pin NRST is asserted, the bit URSTS in RSTC_SR is set. This bit clears only when RSTC_SR is read.

The Reset Controller can also be programmed to generate an interrupt instead of generating a reset. To do so, the bit URSTIEN in RSTC_MR must be written at 1.

14.3.2.2 NRST External Reset Control

The Reset State Manager asserts the signal ext_nreset to assert the NRST pin. When this occurs, the “nrst_out” signal is driven low by the NRST Manager for a time programmed by the field ERSTL in RSTC_MR. This assertion duration, named EXTERNAL_RESET_LENGTH, lasts $2^{(ERSTL+1)}$ Slow Clock cycles. This gives the approximate duration of an assertion between 60 µs and 2 seconds. Note that ERSTL at 0 defines a two-cycle duration for the NRST pulse.

This feature allows the Reset Controller to shape the NRST pin level, and thus to guarantee that the NRST line is driven low for a time compliant with potential external devices connected on the system reset.

As the field is within RSTC_MR, which is backed-up, this field can be used to shape the system power-up reset for devices requiring a longer startup time than the Slow Clock Oscillator.

14.3.3 Reset States

The Reset State Manager handles the different reset sources and generates the internal reset signals. It reports the reset status in the field RSTTYP of the Status Register (RSTC_SR). The update of the field RSTTYP is performed when the processor reset is released.

14.3.3.1 General Reset

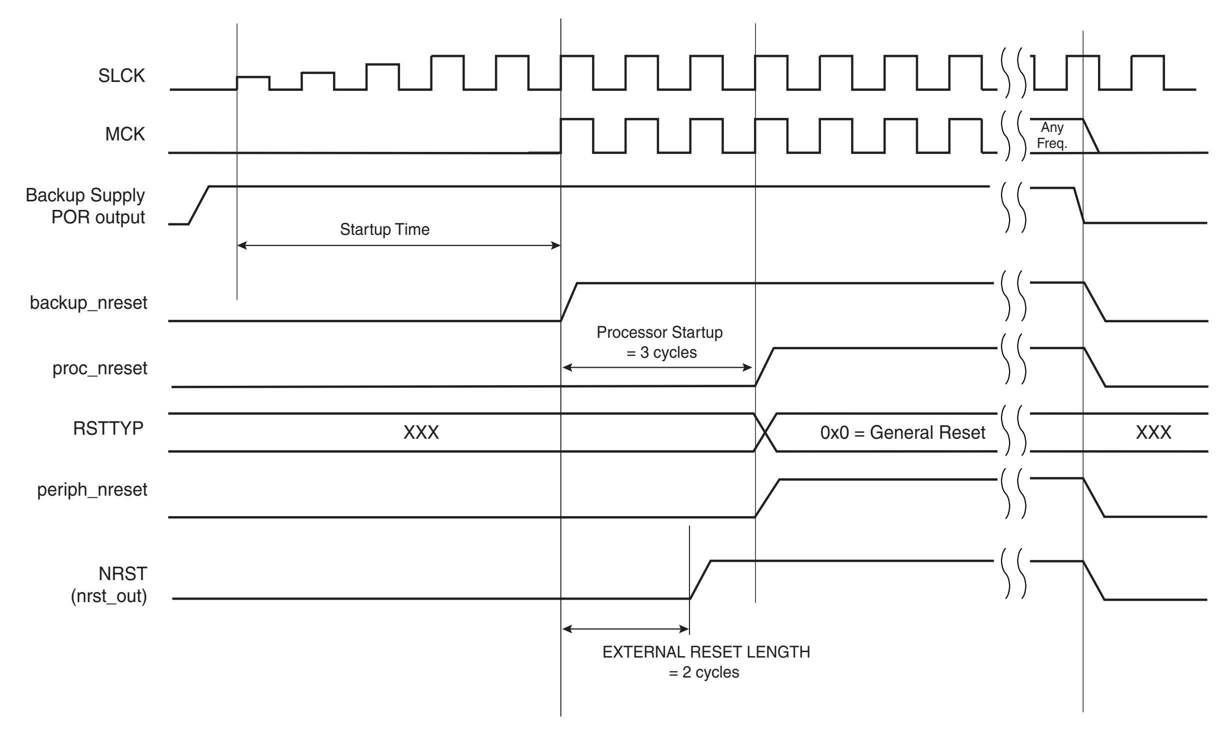
A general reset occurs when VDDBU is powered on. The backup supply POR cell output rises and is filtered with a Startup Counter, which operates at Slow Clock. The purpose of this counter is to make sure the Slow Clock oscillator is stable before starting up the device. The length of startup time is hardcoded to comply with the Slow Clock Oscillator startup time.

After this time, the processor clock is released at Slow Clock and all the other signals remains valid for 2 cycles for proper processor and logic reset. Then, all the reset signals are released and the field RSTTYP in RSTC_SR reports a General Reset. As the RSTC_MR is reset, the NRST line rises 2 cycles after the backup_nreset, as ERSTL defaults at value 0x0.

When VDDBU is detected low by the Backup Supply POR Cell, all resets signals are immediately asserted, even if the Main Supply POR Cell does not report a Main Supply shut down.

Figure 14-3 shows how the General Reset affects the reset signals.

Figure 14-3. General Reset State



14.3.3.2 Wake-up Reset

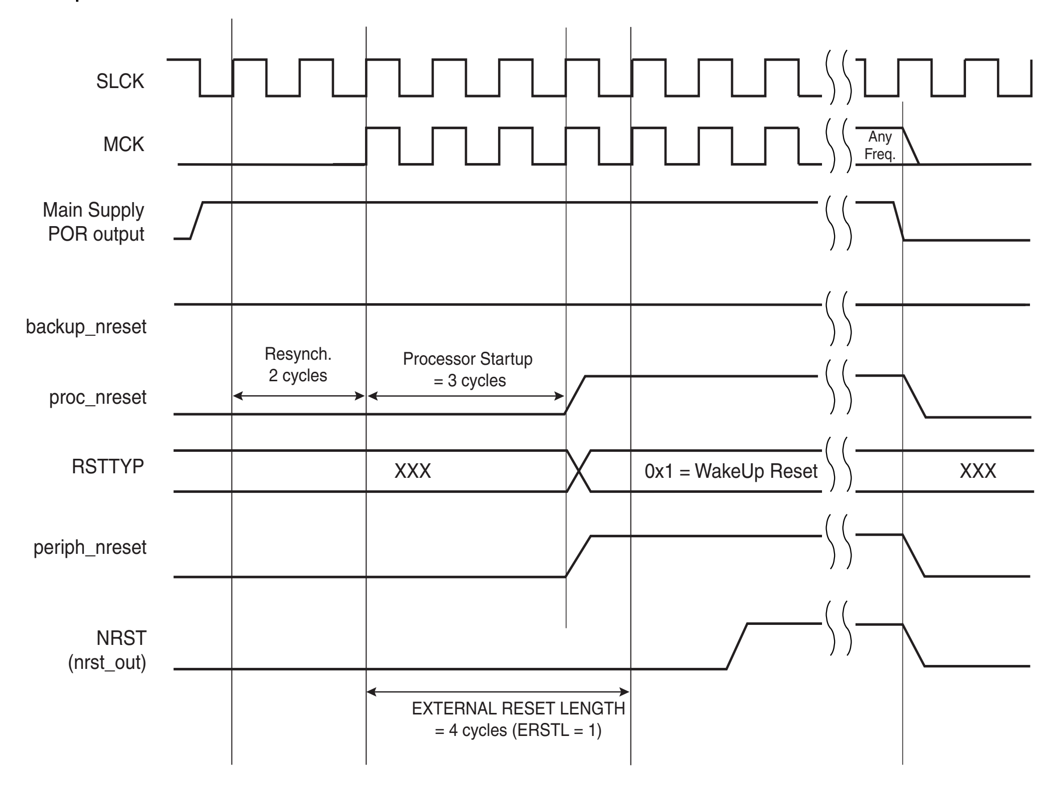
The Wake-up Reset occurs when the Main Supply is down. When the Main Supply POR output is active, all the reset signals are asserted except backup_nreset. When the Main Supply powers up, the POR output is resynchronized on Slow Clock. The processor clock is then re-enabled during 2 Slow Clock cycles, depending on the requirements of the ARM processor.

At the end of this delay, the processor and other reset signals rise. The field RSTTYP in RSTC_SR is updated to report a Wake-up Reset.

The "nrst_out" remains asserted for EXTERNAL_RESET_LENGTH cycles. As RSTC_MR is backed-up, the programmed number of cycles is applicable.

When the Main Supply is detected falling, the reset signals are immediately asserted. This transition is synchronous with the output of the Main Supply POR.

Figure 14-4. Wake-up State



14.3.3.3 User Reset

The User Reset is entered when a low level is detected on the NRST pin and the bit URSTEN in RSTC_MR is at 1. The NRST input signal is resynchronized with SLCK to insure proper behavior of the system.

The User Reset is entered as soon as a low level is detected on NRST. The Processor Reset and the Peripheral Reset are asserted.

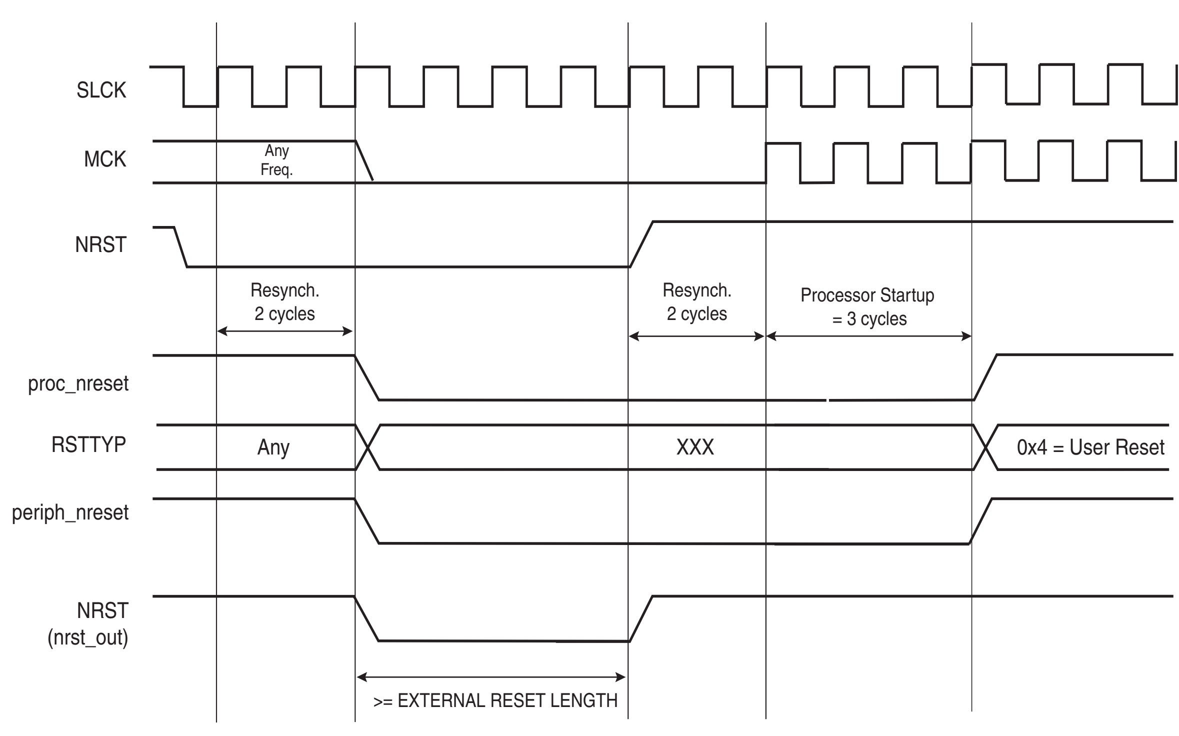
The User Reset is left when NRST rises, after a two-cycle resynchronization time and a three-cycle processor startup. The processor clock is re-enabled as soon as NRST is confirmed high.

When the processor reset signal is released, the RSTTYP field of the Status Register (RSTC_SR) is loaded with the value 0x4, indicating a User Reset.



The NRST Manager guarantees that the NRST line is asserted for EXTERNAL_RESET_LENGTH Slow Clock cycles, as programmed in the field ERSTL. However, if NRST does not rise after EXTERNAL_RESET_LENGTH because it is driven low externally, the internal reset lines remain asserted until NRST actually rises.

Figure 14-5. User Reset State



14.3.3.4 Software Reset

The Reset Controller offers several commands used to assert the different reset signals. These commands are performed by writing the Control Register (RSTC_CR) with the following bits at 1:

- PROCRST: Writing PROCRST at 1 resets the processor and the watchdog timer.
- PERRST: Writing PERRST at 1 resets all the embedded peripherals, including the memory system, and, in particular, the Remap Command. The Peripheral Reset is generally used for debug purposes.
- EXTRST: Writing EXTRST at 1 asserts low the NRST pin during a time defined by the field ERSTL in the Mode Register (RSTC_MR).

The software reset is entered if at least one of these bits is set by the software. All these commands can be performed independently or simultaneously. The software reset lasts 2 Slow Clock cycles.

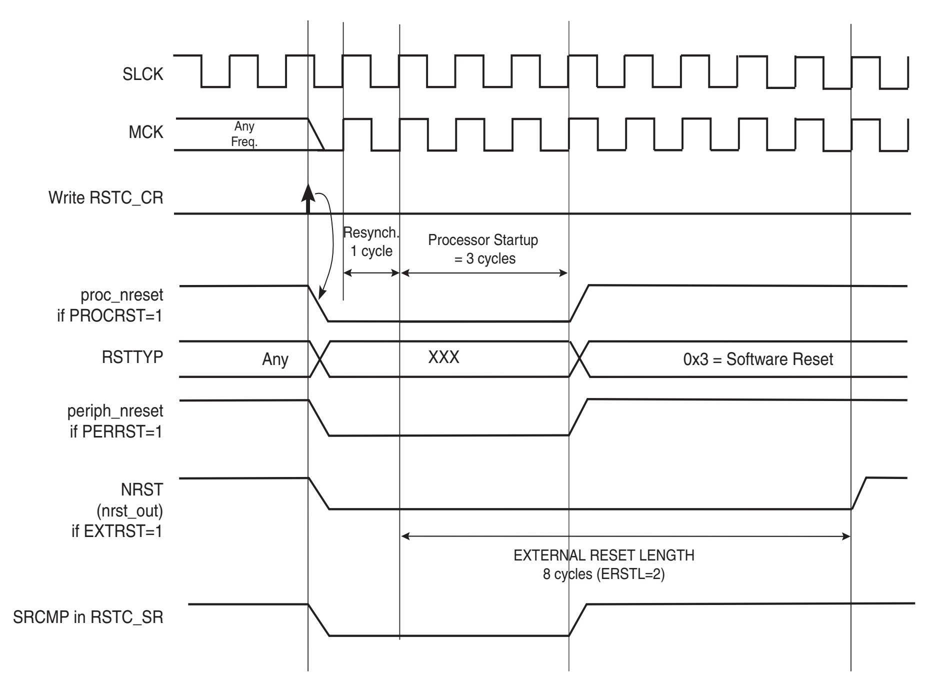
The internal reset signals are asserted as soon as the register write is performed. This is detected on the Master Clock (MCK). They are released when the software reset is left, i.e.; synchronously to SLCK.

If EXTRST is set, the nrst_out signal is asserted depending on the programming of the field ERSTL. However, the resulting falling edge on NRST does not lead to a User Reset.

If and only if the PROCRST bit is set, the Reset Controller reports the software status in the field RSTTYP of the Status Register (RSTC_SR). Other Software Resets are not reported in RSTTYP.

As soon as a software operation is detected, the bit SRCMP (Software Reset Command in Progress) is set in the Status Register (RSTC_SR). It is cleared as soon as the software reset is left. No other software reset can be performed while the SRCMP bit is set, and writing any value in RSTC_CR has no effect.

Figure 14-6. Software Reset



14.3.3.5 Watchdog Reset

The Watchdog Reset is entered when a watchdog fault occurs. This state lasts 2 Slow Clock cycles.

When in Watchdog Reset, assertion of the reset signals depends on the WDRPROC bit in WDT_MR:

- If WDRPROC is 0, the Processor Reset and the Peripheral Reset are asserted. The NRST line is also asserted, depending on the programming of the field ERSTL. However, the resulting low level on NRST does not result in a User Reset state.
- If WDRPROC = 1, only the processor reset is asserted.

The Watchdog Timer is reset by the proc_nreset signal. As the watchdog fault always causes a processor reset if WDRSTEN is set, the Watchdog Timer is always reset after a Watchdog Reset, and the Watchdog is enabled by default and with a period set to a maximum.

When the WDRSTEN in WDT_MR bit is reset, the watchdog fault has no impact on the reset controller.

Figure 14-7. Watchdog Reset

14.3.4 Reset State Priorities

The Reset State Manager manages the following priorities between the different reset sources, given in descending order:

- Backup Reset
- Wake-up Reset
- Watchdog Reset
- Software Reset
- User Reset

Particular cases are listed below:

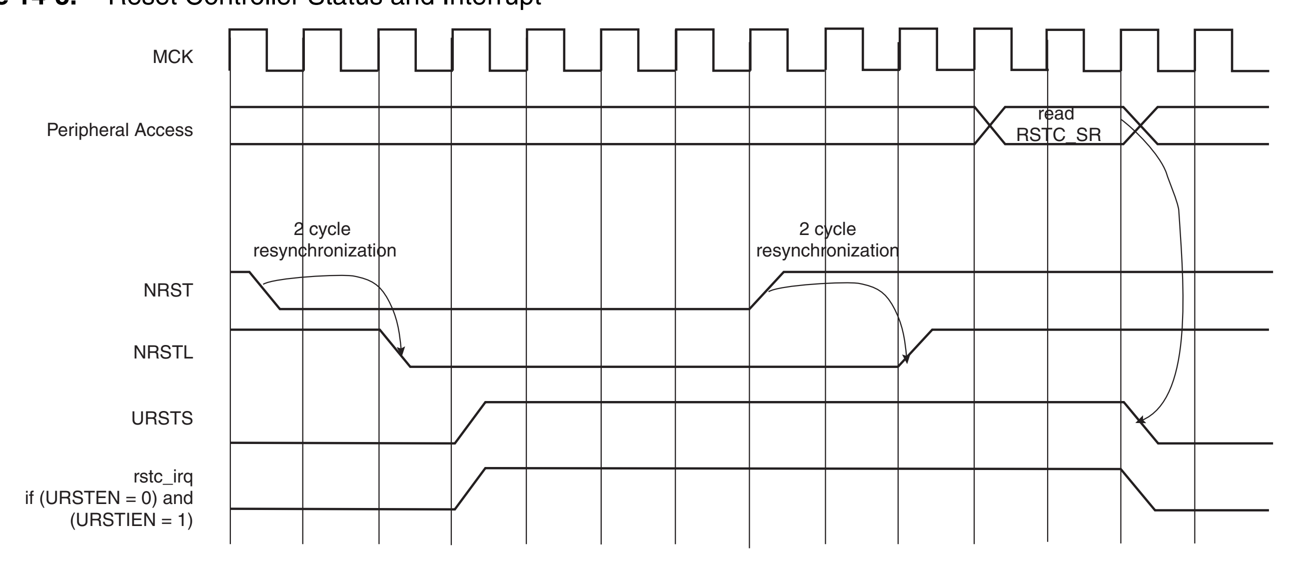
- When in User Reset:
  – A watchdog event is impossible because the Watchdog Timer is being reset by the proc_nreset signal.
  – A software reset is impossible, since the processor reset is being activated.
- When in Software Reset:
  – A watchdog event has priority over the current state.
  – The NRST has no effect.
- When in Watchdog Reset:
  – The processor reset is active and so a Software Reset cannot be programmed.
  – A User Reset cannot be entered.

14.3.5 Reset Controller Status Register

The Reset Controller status register (RSTC_SR) provides several status fields:

- RSTTYP field: This field gives the type of the last reset, as explained in previous sections.
- SRCMP bit: This field indicates that a Software Reset Command is in progress and that no further software reset should be performed until the end of the current one. This bit is automatically cleared at the end of the current software reset.
- NRSTL bit: The NRSTL bit of the Status Register gives the level of the NRST pin sampled on each MCK rising edge.
- URSTS bit: A high-to-low transition of the NRST pin sets the URSTS bit of the RSTC_SR register. This transition is also detected on the Master Clock (MCK) rising edge (see Figure 14-8). If the User Reset is disabled (URSTEN = 0) and if the interruption is enabled by the URSTIEN bit in the RSTC_MR register, the URSTS bit triggers an interrupt. Reading the RSTC_SR status register resets the URSTS bit and clears the interrupt.

Figure 14-8. Reset Controller Status and Interrupt



14.4 Reset Controller (RSTC) User Interface

Table 14-1. Reset Controller (RSTC) Register Mapping

Offset	Register	Name	Access	Reset Value	Back-up Reset Value
0x00	Control Register	RSTC_CR	Write-only	-	
0x04	Status Register	RSTC_SR	Read-only	0x0000_0001	0x0000_0000
0x08	Mode Register	RSTC_MR	Read/Write	-	0x0000_0000

Note: 1. The reset value of RSTC_SR either reports a General Reset or a Wake-up Reset depending on last rising power supply.

14.4.1 Reset Controller Control Register

Register Name: RSTC_CR

Access Type: Write-only

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–		–

7	6	5	4	3	2	1	0
–	–	–	–	EXTRST	PERRST	–	PROCRST

- **PROCRST: Processor Reset**

0 = No effect.

1 = If KEY is correct, resets the processor.

- **PERRST: Peripheral Reset**

0 = No effect.

1 = If KEY is correct, resets the peripherals.

- **EXTRST: External Reset**

0 = No effect.

1 = If KEY is correct, asserts the NRST pin.

- **KEY: Password**

Should be written at value 0xA5. Writing any other value in this field aborts the write operation.

14.4.2 Reset Controller Status Register

Register Name: RSTC_SR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	SRCMP	NRSTL

15	14	13	12	11	10	9	8
–	–	–	–	–	RSTTYP		

7	6	5	4	3	2	1	0
–	–	–	–	–	–		URSTS

- **URSTS: User Reset Status**

0 = No high-to-low edge on NRST happened since the last read of RSTC_SR.

1 = At least one high-to-low transition of NRST has been detected since the last read of RSTC_SR.

- **RSTTYP: Reset Type**

Reports the cause of the last processor reset. Reading this RSTC_SR does not reset this field.

RSTTYP			Reset Type	Comments
0	0	0	General Reset	Both VDDCORE and VDDBU rising
0	0	1	Wake Up Reset	VDDCORE rising
0	1	0	Watchdog Reset	Watchdog fault occurred
0	1	1	Software Reset	Processor reset required by the software
1	0	0	User Reset	NRST pin detected low

- **NRSTL: NRST Pin Level**

Registers the NRST Pin Level at Master Clock (MCK).

- **SRCMP: Software Reset Command in Progress**

0 = No software command is being performed by the reset controller. The reset controller is ready for a software command.

1 = A software reset command is being performed by the reset controller. The reset controller is busy.

14.4.3 Reset Controller Mode Register

Register Name: RSTC_MR

Access Type: Read/Write

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	



15	14	13	12	11	10	9	8
–	–	–	–	ERSTL			

7	6	5	4	3	2	1	0
–	–		URSTIEN	–	–	–	URSTEN

- **URSTEN: User Reset Enable**

0 = The detection of a low level on the pin NRST does not generate a User Reset.

1 = The detection of a low level on the pin NRST triggers a User Reset.

- **URSTIEN: User Reset Interrupt Enable**

0 = USRTS bit in RSTC_SR at 1 has no effect on rstc_irq.

1 = USRTS bit in RSTC_SR at 1 asserts rstc_irq if URSTEN = 0.

- **ERSTL: External Reset Length**

This field defines the external reset length. The external reset is asserted during a time of $2^{(ERSTL+1)}$ Slow Clock cycles. This allows assertion duration to be programmed between 60 µs and 2 seconds.

- **KEY: Password**

Should be written at value 0xA5. Writing any other value in this field aborts the write operation.

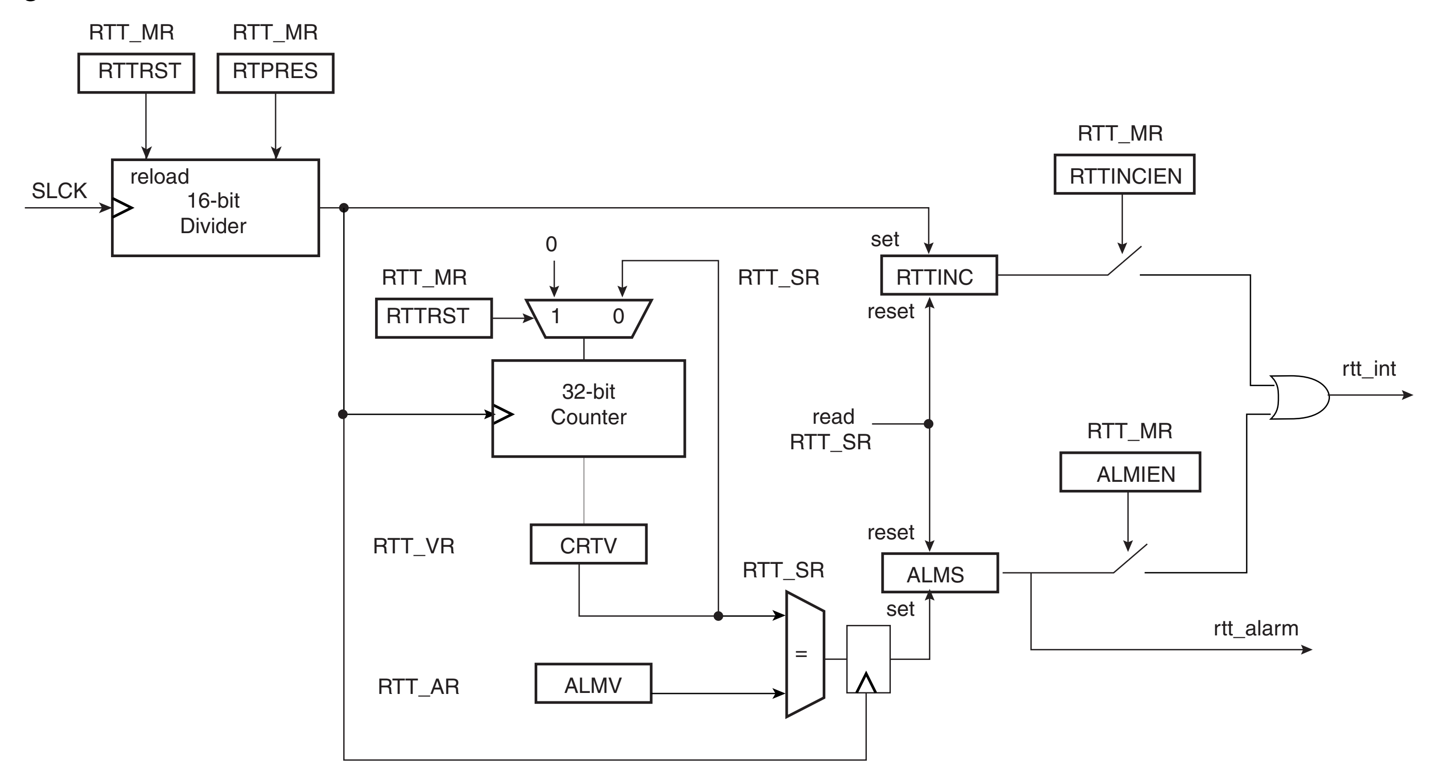
15. Real-time Timer (RTT)

15.1 Overview

The Real-time Timer is built around a 32-bit counter and used to count elapsed seconds. It generates a periodic interrupt or/and triggers an alarm on a programmed value.

15.2 Block Diagram

Figure 15-1. Real-time Timer



15.3 Functional Description

The Real-time Timer is used to count elapsed seconds. It is built around a 32-bit counter fed by Slow Clock divided by a programmable 16-bit value. The value can be programmed in the field RTPRES of the Real-time Mode Register (RTT_MR).

Programming RTPRES at 0x00008000 corresponds to feeding the real-time counter with a 1 Hz signal (if the Slow Clock is 32.768 Hz). The 32-bit counter can count up to 2^{32} seconds, corresponding to more than 136 years, then roll over to 0.

The Real-time Timer can also be used as a free-running timer with a lower time-base. The best accuracy is achieved by writing RTPRES to 3. Programming RTPRES to 1 or 2 is possible, but may result in losing status events because the status register is cleared two Slow Clock cycles after read. Thus if the RTT is configured to trigger an interrupt, the interrupt occurs during 2 Slow Clock cycles after reading RTT_SR. To prevent several executions of the interrupt handler, the interrupt must be disabled in the interrupt handler and re-enabled when the status register is clear.

The Real-time Timer value (CRTV) can be read at any time in the register RTT_VR (Real-time Value Register). As this value can be updated asynchronously from the Master Clock, it is advisable to read this register twice at the same value to improve accuracy of the returned value.

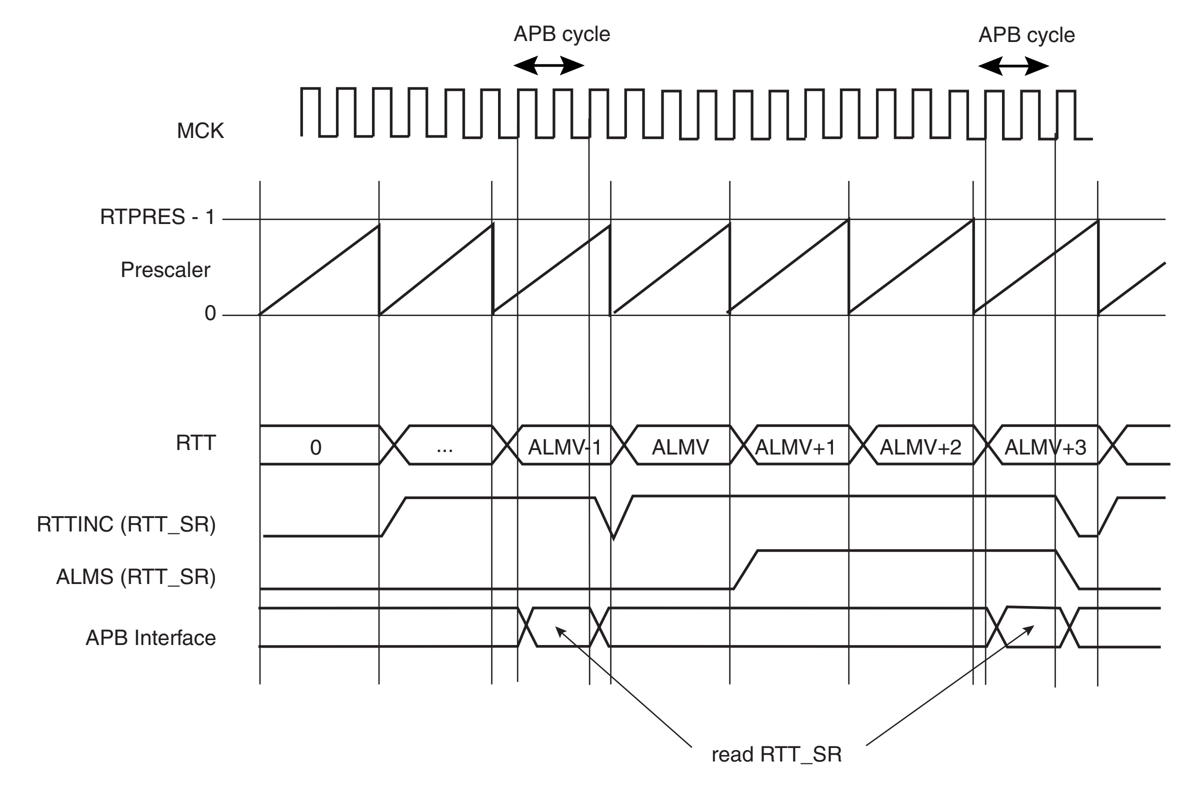
The current value of the counter is compared with the value written in the alarm register RTT_AR (Real-time Alarm Register). If the counter value matches the alarm, the bit ALMS in RTT_SR is set. The alarm register is set to its maximum value, corresponding to 0xFFFF_FFFF, after a reset.

The bit RTTINC in RTT_SR is set each time the Real-time Timer counter is incremented. This bit can be used to start a periodic interrupt, the period being one second when the RTPRES is programmed with 0x8000 and Slow Clock equal to 32.768 Hz.

Reading the RTT_SR status register resets the RTTINC and ALMS fields.

Writing the bit RTTRST in RTT_MR immediately reloads and restarts the clock divider with the new programmed value. This also resets the 32-bit counter.

Figure 15-2. RTT Counting



15.4 Real-time Timer (RTT) User Interface

Table 15-1. Real-time Timer Register Mapping

Offset	Register	Name	Access	Reset Value
0x00	Mode Register	RTT_MR	Read/Write	0x0000_8000
0x04	Alarm Register	RTT_AR	Read/Write	0xFFFF_FFFF
0x08	Value Register	RTT_VR	Read-only	0x0000_0000
0x0C	Status Register	RTT_SR	Read-only	0x0000_0000

15.4.1 Real-time Timer Mode Register

Register Name: RTT_MR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	RTTRST	RTTINCIEN	ALMIEN

15	14	13	12	11	10	9	8
RTPRES							

7	6	5	4	3	2	1	0
RTPRES							

- **RTPRES: Real-time Timer Prescaler Value**

Defines the number of SLCK periods required to increment the real-time timer. RTPRES is defined as follows:

RTPRES = 0: The Prescaler Period is equal to 2^{16}

RTPRES $\neq$ 0: The Prescaler Period is equal to RTPRES.

- **ALMIEN: Alarm Interrupt Enable**

0 = The bit ALMS in RTT_SR has no effect on interrupt.

1 = The bit ALMS in RTT_SR asserts interrupt.

- **RTTINCIEN: Real-time Timer Increment Interrupt Enable**

0 = The bit RTTINC in RTT_SR has no effect on interrupt.

1 = The bit RTTINC in RTT_SR asserts interrupt.

- **RTTRST: Real-time Timer Restart**

1 = Reloads and restarts the clock divider with the new programmed value. This also resets the 32-bit counter.

15.4.2 Real-time Timer Alarm Register

Register Name: RTT_AR

Access Type: Read/Write

31	30	29	28	27	26	25	24
ALMV							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ALMV							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ALMV							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ALMV							

- **ALMV: Alarm Value**

Defines the alarm value (ALMV+1) compared with the Real-time Timer.

15.4.3 Real-time Timer Value Register

Register Name: RTT_VR

Access Type: Read-only

31	30	29	28	27	26	25	24
CRTV							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CRTV							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CRTV							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CRTV							

- **CRTV: Current Real-time Value**

Returns the current value of the Real-time Timer.

15.4.4 Real-time Timer Status Register

Register Name: RTT_SR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	RTTINC	ALMS

- **ALMS: Real-time Alarm Status**

0 = The Real-time Alarm has not occurred since the last read of RTT_SR.

1 = The Real-time Alarm occurred since the last read of RTT_SR.

- **RTTINC: Real-time Timer Increment**

0 = The Real-time Timer has not been incremented since the last read of the RTT_SR.

1 = The Real-time Timer has been incremented since the last read of the RTT_SR.

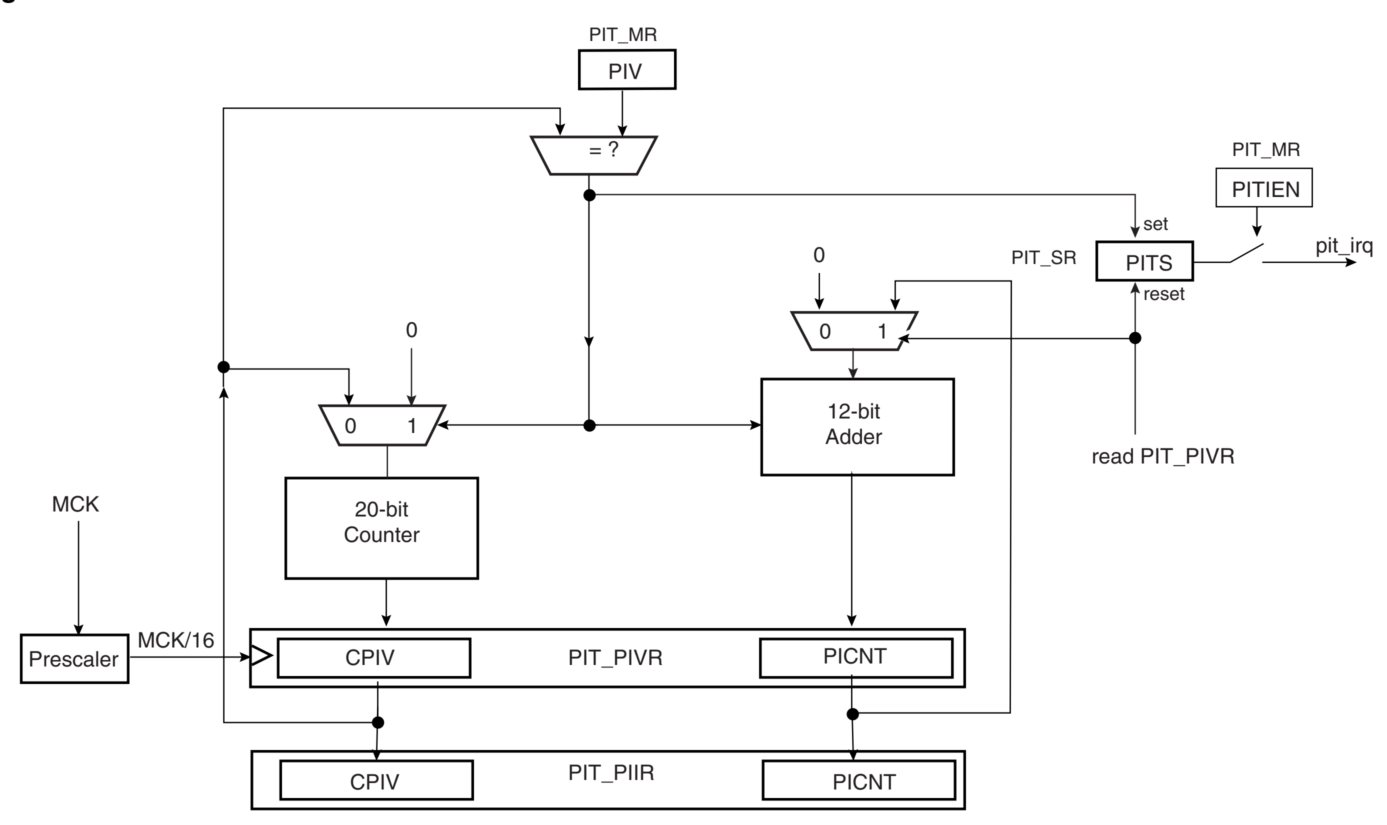
16. Periodic Interval Timer (PIT)

16.1 Overview

The Periodic Interval Timer (PIT) provides the operating system's scheduler interrupt. It is designed to offer maximum accuracy and efficient management, even for systems with long response time.

16.2 Block Diagram

Figure 16-1. Periodic Interval Timer



16.3 Functional Description

The Periodic Interval Timer aims at providing periodic interrupts for use by operating systems.

The PIT provides a programmable overflow counter and a reset-on-read feature. It is built around two counters: a 20-bit CPIV counter and a 12-bit PICNT counter. Both counters work at Master Clock /16.

The first 20-bit CPIV counter increments from 0 up to a programmable overflow value set in the field PIV of the Mode Register (PIT_MR). When the counter CPIV reaches this value, it resets to 0 and increments the Periodic Interval Counter, PICNT. The status bit PITS in the Status Register (PIT_SR) rises and triggers an interrupt, provided the interrupt is enabled (PITIEN in PIT_MR).

Writing a new PIV value in PIT_MR does not reset/restart the counters.

When CPIV and PICNT values are obtained by reading the Periodic Interval Value Register (PIT_PIVR), the overflow counter (PICNT) is reset and the PITS is cleared, thus acknowledging the interrupt. The value of PICNT gives the number of periodic intervals elapsed since the last read of PIT_PIVR.

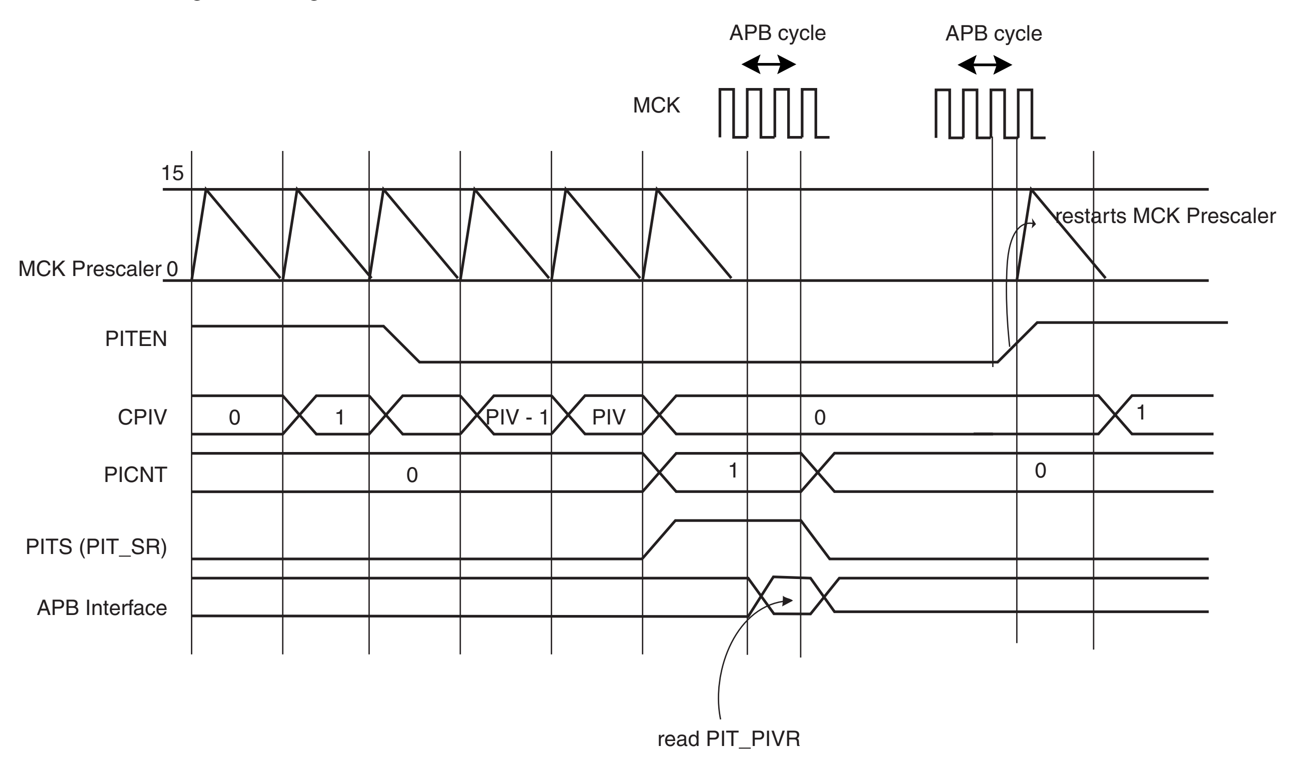
When CPIV and PICNT values are obtained by reading the Periodic Interval Image Register (PIT_PIIR), there is no effect on the counters CPIV and PICNT, nor on the bit PITS. For example, a profiler can read PIT_PIIR without clearing any pending interrupt, whereas a timer interrupt clears the interrupt by reading PIT_PIVR.

The PIT may be enabled/disabled using the PITEN bit in the PIT_MR register (disabled on reset). The PITEN bit only becomes effective when the CPIV value is 0. Figure 16-2 illustrates the PIT counting. After the PIT Enable bit is reset (PITEN= 0), the CPIV goes on counting until the PIV value is reached, and is then reset. PIT restarts counting, only if the PITEN is set again.

The PIT is stopped when the core enters debug state.



Figure 16-2. Enabling/Disabling PIT with PITEN



16.4 Periodic Interval Timer (PIT) User Interface

Table 16-1. Periodic Interval Timer (PIT) Register Mapping

Offset	Register	Name	Access	Reset Value
0x00	Mode Register	PIT_MR	Read/Write	0x000F_FFFF
0x04	Status Register	PIT_SR	Read-only	0x0000_0000
0x08	Periodic Interval Value Register	PIT_PIVR	Read-only	0x0000_0000
0x0C	Periodic Interval Image Register	PIT_PIIR	Read-only	0x0000_0000

16.4.1 Periodic Interval Timer Mode Register

Register Name: PIT_MR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	PITIEN	PITEN

23	22	21	20	19	18	17	16
–	–	–	–	PIV			

15	14	13	12	11	10	9	8
PIV							

7	6	5	4	3	2	1	0
PIV							

- **PIV: Periodic Interval Value**

Defines the value compared with the primary 20-bit counter of the Periodic Interval Timer (CPIV). The period is equal to (PIV + 1).

- **PITEN: Period Interval Timer Enabled**

0 = The Periodic Interval Timer is disabled when the PIV value is reached.

1 = The Periodic Interval Timer is enabled.

- **PITIEN: Periodic Interval Timer Interrupt Enable**

0 = The bit PITS in PIT_SR has no effect on interrupt.

1 = The bit PITS in PIT_SR asserts interrupt.

16.4.2 Periodic Interval Timer Status Register

Register Name: PIT_SR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	PITS

- **PITS: Periodic Interval Timer Status**

0 = The Periodic Interval timer has not reached PIV since the last read of PIT_PIVR.

1 = The Periodic Interval timer has reached PIV since the last read of PIT_PIVR.

16.4.3 Periodic Interval Timer Value Register

Register Name: PIT_PIVR

Access Type: Read-only

31	30	29	28	27	26	25	24
PICNT							

23	22	21	20	19	18	17	16
PICNT				CPIV			

15	14	13	12	11	10	9	8
CPIV							

7	6	5	4	3	2	1	0
CPIV							

Reading this register clears PITS in PIT_SR.

- **CPIV: Current Periodic Interval Value**

Returns the current value of the periodic interval timer.

- **PICNT: Periodic Interval Counter**

Returns the number of occurrences of periodic intervals since the last read of PIT_PIVR.

16.4.4 Periodic Interval Timer Image Register

Register Name: PIT_PIIR

Access Type: Read-only

31	30	29	28	27	26	25	24
PICNT							

23	22	21	20	19	18	17	16
PICNT				CPIV			

15	14	13	12	11	10	9	8
CPIV							

7	6	5	4	3	2	1	0
CPIV							

- **CPIV: Current Periodic Interval Value**

Returns the current value of the periodic interval timer.

- **PICNT: Periodic Interval Counter**

Returns the number of occurrences of periodic intervals since the last read of PIT_PIVR.

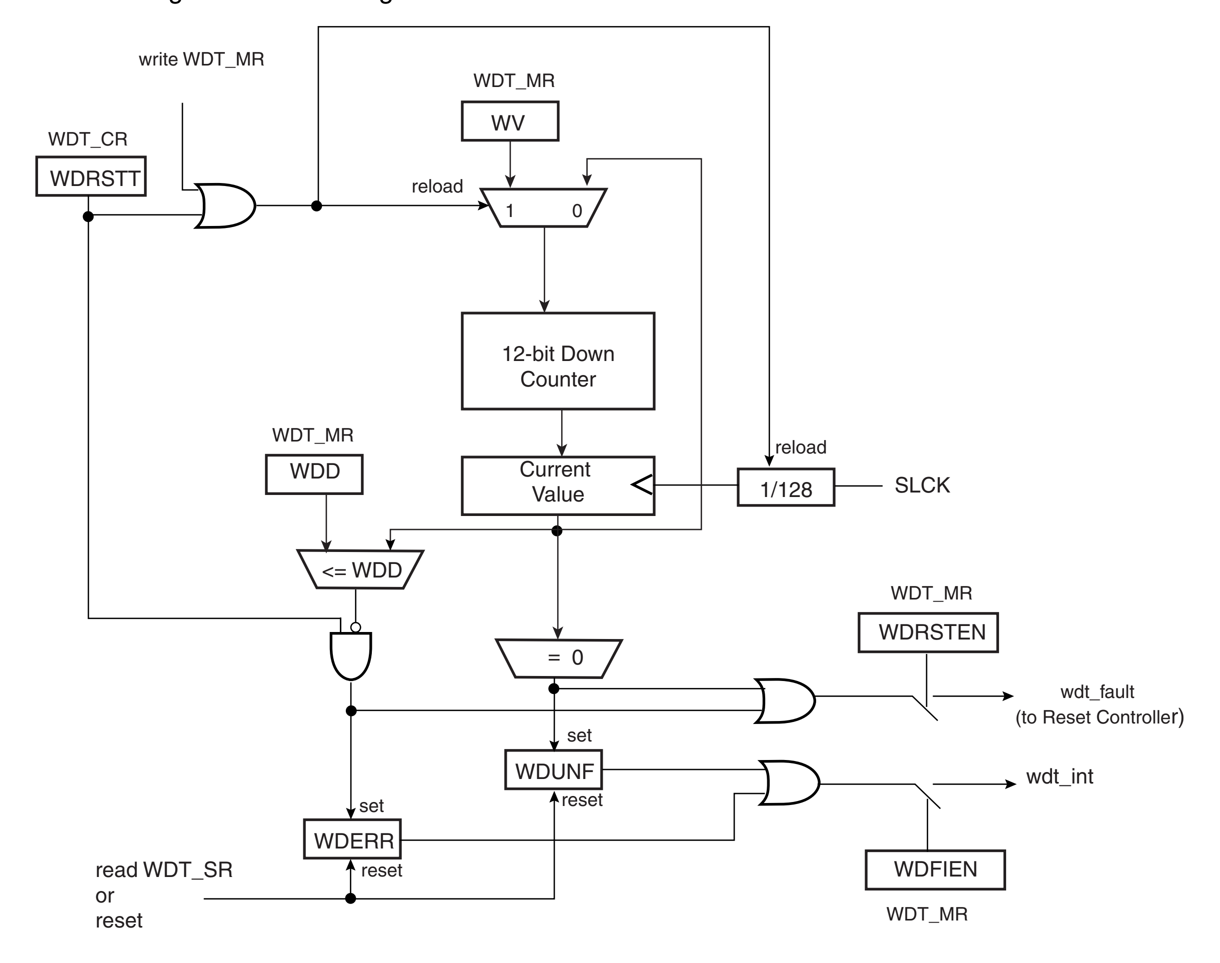
17. Watchdog Timer (WDT)

17.1 Overview

The Watchdog Timer can be used to prevent system lock-up if the software becomes trapped in a deadlock. It features a 12-bit down counter that allows a watchdog period of up to 16 seconds (slow clock at 32.768 kHz). It can generate a general reset or a processor reset only. In addition, it can be stopped while the processor is in debug mode or idle mode.

17.2 Block Diagram

Figure 17-1. Watchdog Timer Block Diagram



17.3 Functional Description

The Watchdog Timer can be used to prevent system lock-up if the software becomes trapped in a deadlock. It is supplied with VDDCORE. It restarts with initial values on processor reset.

The Watchdog is built around a 12-bit down counter, which is loaded with the value defined in the field WV of the Mode Register (WDT_MR). The Watchdog Timer uses the Slow Clock divided by 128 to establish the maximum Watchdog period to be 16 seconds (with a typical Slow Clock of 32.768 kHz).

After a Processor Reset, the value of WV is 0xFFF, corresponding to the maximum value of the counter with the external reset generation enabled (field WDRSTEN at 1 after a Backup Reset). This means that a default Watchdog is running at reset, i.e., at power-up. The user must either disable it (by setting the WDDIS bit in WDT_MR) if he does not expect to use it or must reprogram it to meet the maximum Watchdog period the application requires.

The Watchdog Mode Register (WDT_MR) can be written only once. Only a processor reset resets it. Writing the WDT_MR register reloads the timer with the newly programmed mode parameters.

In normal operation, the user reloads the Watchdog at regular intervals before the timer underflow occurs, by writing the Control Register (WDT_CR) with the bit WDRSTT to 1. The Watchdog counter is then immediately reloaded from WDT_MR and restarted, and the Slow Clock 128 divider is reset and restarted. The WDT_CR register is write-protected. As a result, writing WDT_CR without the correct hard-coded key has no effect. If an underflow does occur, the "wdt_fault" signal to the Reset Controller is asserted if the bit WDRSTEN is set in the Mode Register (WDT_MR). Moreover, the bit WDUNF is set in the Watchdog Status Register (WDT_SR).

To prevent a software deadlock that continuously triggers the Watchdog, the reload of the Watchdog must occur in a window defined by 0 and WDD in the WDT_MR:

$0 \leq WDT \leq WDD$; writing WDRSTT restarts the Watchdog Timer.

Any attempt to restart the Watchdog Timer in the range [WDV; WDD] results in a Watchdog error, even if the Watchdog is disabled. The bit WDERR is updated in the WDT_SR and the "wdt_fault" signal to the Reset Controller is asserted.

Note that this feature can be disabled by programming a WDD value greater than or equal to the WDV value. In such a configuration, restarting the Watchdog Timer is permitted in the whole range [0; WDV] and does not generate an error. This is the default configuration on reset (the WDD and WDV values are equal).

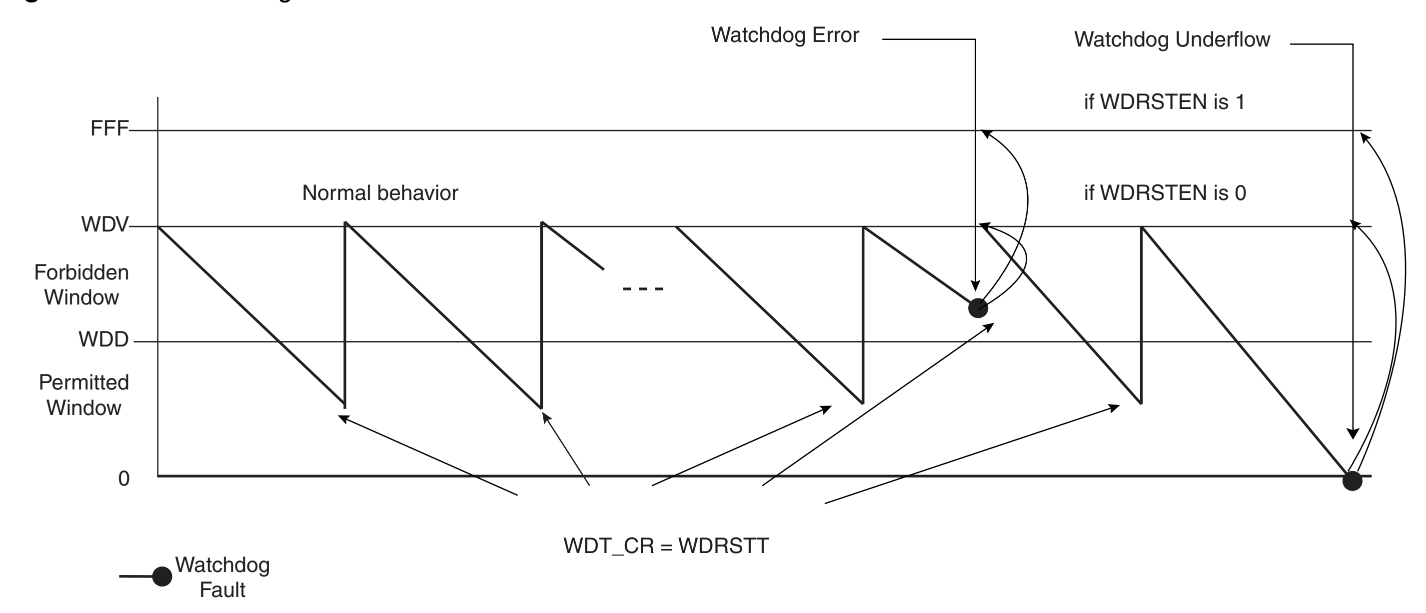
The status bits WDUNF (Watchdog Underflow) and WDERR (Watchdog Error) trigger an interrupt, provided the bit WDFIEN is set in the mode register. The signal "wdt_fault" to the reset controller causes a Watchdog reset if the WDRSTEN bit is set as already explained in the reset controller programmer Datasheet. In that case, the processor and the Watchdog Timer are reset, and the WDERR and WDUNF flags are reset.

If a reset is generated or if WDT_SR is read, the status bits are reset, the interrupt is cleared, and the "wdt_fault" signal to the reset controller is deasserted.

Writing the WDT_MR reloads and restarts the down counter.

While the processor is in debug state or in idle mode, the counter may be stopped depending on the value programmed for the bits WDIDLEHLT and WDDBGHLT in the WDT_MR.

Figure 17-2. Watchdog Behavior



17.4 Watchdog Timer (WDT) User Interface

Table 17-1. Watchdog Timer (WDT) Register Mapping

Offset	Register	Name	Access	Reset Value
0x00	Control Register	WDT_CR	Write-only	-
0x04	Mode Register	WDT_MR	Read/Write Once	0x3FFF_2FFF
0x08	Status Register	WDT_SR	Read-only	0x0000_0000

17.4.1 Watchdog Timer Control Register

Register Name: WDT_CR

Access Type: Write-only

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WDRSTT

- **WDRSTT: Watchdog Restart**

0: No effect.

1: Restarts the Watchdog.

- **KEY: Password**

Should be written at value 0xA5. Writing any other value in this field aborts the write operation.

17.4.2 Watchdog Timer Mode Register

Register Name: WDT_MR

Access Type: Read / Write Once

31	30	29	28	27	26	25	24
		WDIDLEHLT	WDDBGHLT	WDD			

23	22	21	20	19	18	17	16
WDD							

15	14	13	12	11	10	9	8
WDDIS	WDRPROC	WDRSTEN	WDFIEN	WDV			

7	6	5	4	3	2	1	0
WDV							

- **WDV: Watchdog Counter Value**

Defines the value loaded in the 12-bit Watchdog Counter.

- **WDFIEN: Watchdog Fault Interrupt Enable**

0: A Watchdog fault (underflow or error) has no effect on interrupt.

1: A Watchdog fault (underflow or error) asserts interrupt.

- **WDRSTEN: Watchdog Reset Enable**

0: A Watchdog fault (underflow or error) has no effect on the resets.

1: A Watchdog fault (underflow or error) triggers a Watchdog reset.

- **WDRPROC: Watchdog Reset Processor**

0: If WDRSTEN is 1, a Watchdog fault (underflow or error) activates all resets.

1: If WDRSTEN is 1, a Watchdog fault (underflow or error) activates the processor reset.

- **WDD: Watchdog Delta Value**

Defines the permitted range for reloading the Watchdog Timer. If the Watchdog Timer value is less than or equal to WDD, writing WDT_CR with WDRSTT = 1 restarts the timer. If the Watchdog Timer value is greater than WDD, writing WDT_CR with WDRSTT = 1 causes a Watchdog error.

- **WDDBGHLT: Watchdog Debug Halt**

0: The Watchdog runs when the processor is in debug state.

1: The Watchdog stops when the processor is in debug state.

- **WDIDLEHLT: Watchdog Idle Halt**

0: The Watchdog runs when the system is in idle mode.

1: The Watchdog stops when the system is in idle state.

- **WDDIS: Watchdog Disable**

0: Enables the Watchdog Timer.

1: Disables the Watchdog Timer.

17.4.3 Watchdog Timer Status Register

Register Name: WDT_SR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	WDERR	WDUNF

- **WDUNF: Watchdog Underflow**

0: No Watchdog underflow occurred since the last read of WDT_SR.

1: At least one Watchdog underflow occurred since the last read of WDT_SR.

- **WDERR: Watchdog Error**

0: No Watchdog error occurred since the last read of WDT_SR.

1: At least one Watchdog error occurred since the last read of WDT_SR.

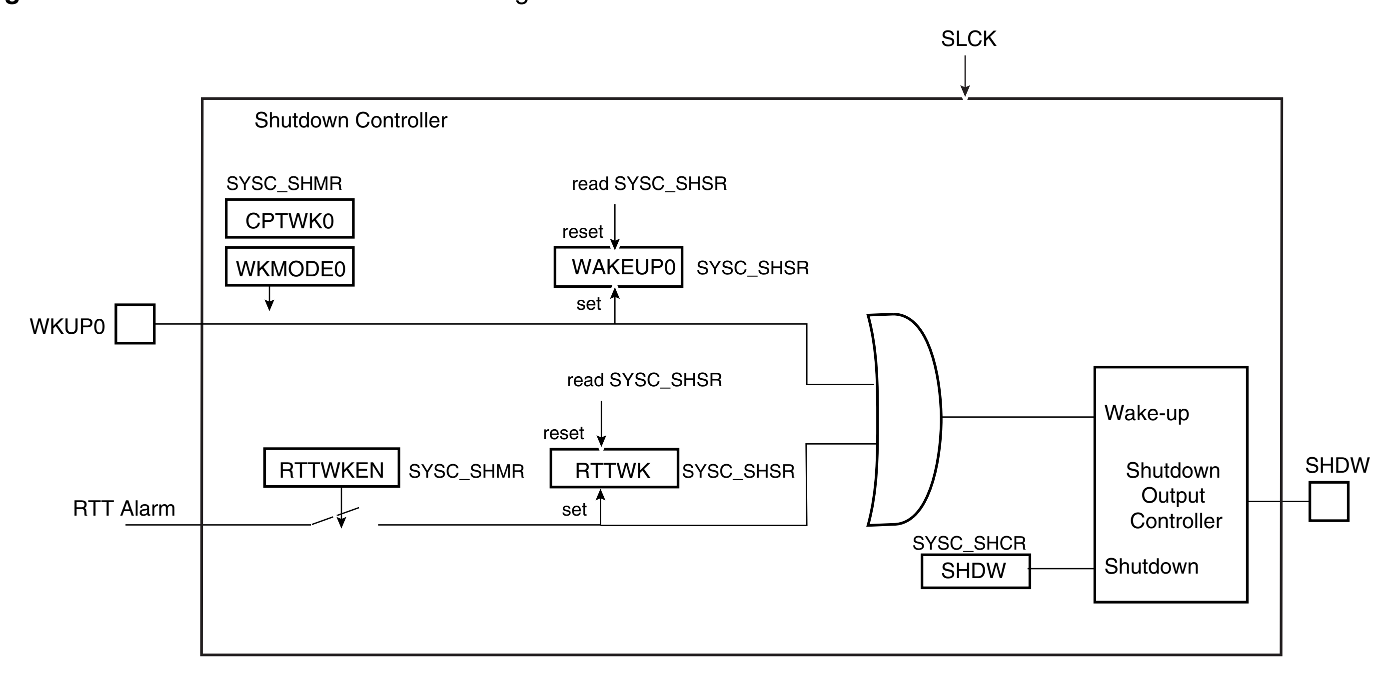
18. Shutdown Controller (SHDWC)

18.1 Description

The Shutdown Controller controls the power supplies VDDIO and VDDCORE and the wake-up detection on debounced input lines.

18.2 Block Diagram

Figure 18-1. Shutdown Controller Block Diagram



18.3 I/O Lines Description

Table 18-1. I/O Lines Description

Name	Description	Type
WKUP0	Wake-up 0 input	Input
SHDW	Shutdown output	Output

18.4 Product Dependencies

18.4.1 Power Management

The Shutdown Controller is continuously clocked by Slow Clock. The Power Management Controller has no effect on the behavior of the Shutdown Controller.

18.5 Functional Description

The Shutdown Controller manages the main power supply. To do so, it is supplied with VDDBU and manages wake-up input pins and one output pin, SHDW.

A typical application connects the pin SHDW to the shutdown input of the DC/DC Converter providing the main power supplies of the system, and especially VDDCORE and/or VDDIO. The wake-up inputs (WKUP0) connect to any push-buttons or signal that wake up the system.

The software is able to control the pin SHDW by writing the Shutdown Control Register (SHDW_CR) with the bit SHDW at 1. This register is password-protected and so the value written should contain the correct key for the command to be taken into account. As a result, the system should be powered down.

A level change on WKUP0 is used as wake-up. Wake-up is configured in the Shutdown Mode Register (SHDW_MR). The transition detector can be programmed to detect either a positive or negative transition or any level change on WKUP0. The detection can also be disabled. Programming is performed by defining WKMODE0.

Moreover, a debouncing circuit can be programmed for WKUP0. The debouncing circuit filters pulses on WKUP0 shorter than the programmed number of 16 SLCK cycles in CPTWK0 of the SHDW_MR register. If the programmed level change is detected on a pin, a counter starts. When the counter reaches the value programmed in the corresponding field, CPTWK0 , the SHDW pin is released. If a new input change is detected before the counter reaches the corresponding value, the counter is stopped and cleared. WAKEUP0 of the Status Register (SHDW_SR) reports the detection of the programmed events on WKUP0, with a reset after the read of SHDW_SR.

18.6 Shutdown Controller (SHDWC) User Interface

18.6.1 Register Mapping

Table 18-2. Shutdown Controller (SHDWC) Registers

Offset	Register	Name	Access	Reset Value
0x00	Shutdown Control Register	SHDW_CR	Write-only	-
0x04	Shutdown Mode Register	SHDW_MR	Read-Write	0x0000_0103
0x18	Shutdown Status Register	SHDW_SR	Read-only	0x0000_0000

18.6.2 Shutdown Control Register

Register Name: SHDW_CR

Access Type: Write-only

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SHDW

- **SHDW: Shut Down Command**

0 = No effect.

1 = If KEY is correct, asserts the SHDW pin.

- **KEY: Password**

Should be written at value 0xA5. Writing any other value in this field aborts the write operation.

18.6.3 Shutdown Mode Register

Register Name: SHDW_MR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–				–	–	–	

7	6	5	4	3	2	1	0
CPTWK0				–	–	WKMODE0	

- **WKMODE0: Wake-up Mode 0**

WKMODE[1:0]		Wake-up Input Transition Selection
0	0	None. No detection is performed on the wake-up input
0	1	Low to high level
1	0	High to low level
1	1	Both levels change

- **CPTWK0: Counter on Wake-up 0**

Defines the number of 16 Slow Clock cycles, the level detection on the corresponding input pin shall last before the wake-up event occurs. Because of the internal synchronization of WKUP0 , the SHDW pin is released (CPTWK x 16 + 1) Slow Clock cycles after the event on WKUP.

18.6.4 Shutdown Status Register

Register Name: SHDW_SR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WAKEUP0

- **WAKEUP0: Wake-up 0 Status**

0 = No wake-up event occurred on the corresponding wake-up input since the last read of SHDW_SR.

1 = At least one wake-up event occurred on the corresponding wake-up input since the last read of SHDW_SR.

19. Bus Matrix

19.1 Overview

The Bus Matrix implements a multi-layer AHB, based on AHB-Lite protocol, that enables parallel access paths between multiple AHB masters and slaves in a system, thus increasing the overall bandwidth. The Bus Matrix interconnects 5 AHB Masters to 5 AHB Slaves. The Bus Matrix user interface is compliant with the ARM Advanced Peripheral Bus and provides 5 Special Function Registers (MATRIX_SFR) that allow the Bus Matrix to support application-specific features.

19.2 Memory Mapping

The Bus Matrix provides one decoder for every AHB Master Interface. The decoder offers each AHB Master several memory mappings. Depending on the product, each memory area may be assigned to several slaves. Booting at the same address while using different AHB slaves (i.e., external RAM, internal ROM, internal Flash, etc.) becomes possible.

The Bus Matrix user interface provides a Master Configuration Register (MATRIX_MCFG) that performs a remap action for every master independently.

19.3 Special Bus Granting Techniques

The Bus Matrix provides some speculative bus granting techniques in order to anticipate access requests from some masters. This technique reduces latency at first accesses. The bus granting technique sets a default master for every slave. At the end of the current access, if no other request is pending, the slave remains connected to its associated default master. A slave can be associated with three kinds of default masters; no default master, last access master and fixed default master.

19.3.1 No Default Master

At the end of current access, if no other request is pending, the slave is disconnected from all masters. No Default Master suits low-power mode.

19.3.2 Last Access Master

At the end of current access, if no other request is pending, the slave remains connected to the last master that performs an access request.

19.3.3 Fixed Default Master

At the end of current access, if no other request is pending, the slave remains connected to its fixed default master. Unlike last access master, the fixed master does not change unless the user changes it by a software action.

To change from one kind of default master to another, the Bus Matrix user interface provides 5 Slave Configuration Registers, one for each slave, that set default master for each slave. The Slave Configuration Register contains two fields; DEFMSTR_TYPE and FIXED_DEFMSTR. The 2-bit DEFMSTR_TYPE flag selects the default master type (no default, last access master, fixed default master) whereas the 3-bit FIXED_DEFMSTR flag selects a fixed default master provided that DEFMSTR_TYPE is set to a fixed default master. See ”Bus Matrix User Interface” on page 124.

19.4 Arbitration

The Bus Matrix provides an arbitration function that reduces latency when conflicting cases occur, i.e., when two or more masters try to access the same slave at the same time. The Bus Matrix arbitration mechanism uses slightly modified round-robin algorithms that grant the bus for the first access to a certain master depending on parameters located in the slave's Slave Configuration Register.

There are three round-robin algorithm types:

- Round-Robin arbitration without default master
- Round-Robin arbitration with last access master
- Round-Robin arbitration with fixed default master

19.4.1 Round-Robin Arbitration Without Default Master

This is the main algorithm used by Bus Matrix arbiters. It allows the Bus Matrix to dispatch requests from different masters to the same slave in a pure round-robin manner. At the end of the current access, if no other request is pending, the slave is disconnected from all masters. This configuration incurs one latency cycle for the first access. Arbitration without default master can be used for masters that perform significant bursts.

19.4.2 Round-Robin Arbitration With Last Access Master

This is a biased round-robin algorithm used by Bus Matrix arbiters. It allows the Bus Matrix to remove one latency cycle for the last master that accessed the slave. In fact, at the end of the current transfer, if no other master request is pending, the slave remains connected to the last master that performs the access. Other non-privileged masters still obtain one latency cycle if they want to access the same slave. This technique can be used for masters that perform mainly single accesses.

19.4.3 Round-Robin Arbitration With Fixed Default Master

This is a biased round-robin algorithm. It allows the Bus Matrix arbiters to remove one latency cycle for the fixed master of a slave. At the end of the current access, the slave remains connected to its fixed default master. Any request attempted by this fixed default master does not cause any latency, whereas other non-privileged masters still obtain one latency cycle. This technique can be used for masters that perform mainly single accesses.



19.5 Bus Matrix User Interface

Table 19-1. Register Mapping

Offset	Register	Name	Access	Reset Value
0x0000	Master Configuration Register	MATRIX_MCFG	Write only	0x00000000
0x0004	Slave Configuration Register 0	MATRIX_SCFG0	Read/Write	0x00000010
0x0008	Slave Configuration Register 1	MATRIX_SCFG1	Read/Write	0x00000010
0x000C	Slave Configuration Register 2	MATRIX_SCFG2	Read/Write	0x00000010
0x0010	Slave Configuration Register 3	MATRIX_SCFG3	Read/Write	0x00000010
0x0014	Slave Configuration Register 4	MATRIX_SCFG4	Read/Write	0x00000010
0x0018 - 0x0020	Reserved	–	–	–
0x0024	MATRIX TCM Configuration Register	MATRIX_TCR	Read/Write	0x00000000
0x028 - 0x002C	Reserved	–	–	–
0x0030	EBI Chip Select Assignment Register	EBI_CSA	Read/Write	0x00000000
0x0034	USB Pad Pull-up Control Register	USB_PUCR	Read/Write	0x00000000

19.5.1 Bus Matrix Master Configuration Register

Register Name: MATRIX_MCFG

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	RCB4	RCB3	RCB2	RCB1	RCB0

- **RCBx: Remap Command Bit for AHB Master x**

0: No effect

1: This Command Bit acts on a toggle basis: writing a 1 alternatively cancels and restores the remapping of addressed slaves from master x.

19.5.2 Bus Matrix Slave Configuration Registers

Register Name: MATRIX_SCFG0...MATRIX_SCFG4

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–			FIXED_DEFMSTR			DEFMSTR_TYPE	

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
SLOT_CYCLE							

- **SLOT_CYCLE: Maximum Number of Allowed Cycles for a Burst**

When the SLOT_CYCLE limit is reached for a burst, it may be broken by another master trying to access this slave.

This limit has been set to avoid locking very slow slaves when very long bursts are used.

This limit should not be very small. An unreasonably small value breaks every burst and the Bus Matrix spends its time arbitrating without performing any data transfer. 16 cycles is a reasonable value for SLOT_CYCLE.

- **DEFMASTR_TYPE: Default Master Type**

0: No Default Master

At the end of the current slave access, if no other master request is pending, the slave is disconnected from all masters.

This results in one cycle latency for the first transfer of a burst.

1: Last Default Master

At the end of the current slave access, if no other master request is pending, the slave remains connected to the last master that accessed it.

This results in not having the one cycle latency when the last master is trying to access the slave again.

2: Fixed Default Master

At the end of the current slave access, if no other master request is pending, the slave connects with the fixed master that has its index in FIXED_DEFMSTR register.

This results in not having the one cycle latency when the fixed master is trying to access the slave again.

- **FIXED_DEFMSTR: Fixed Index of Default Master**

This is the index of the Fixed Default Master for this slave.

19.5.3 Bus Matrix TCM Configuration Register

Register Name: MATRIX_TCR

Access Type: Read/Write

Reset: 0x0000_0000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
DTCM_SIZE				ITCM_SIZE			

- **ITCM_SIZE: Size of ITCM enabled memory block**

0000 0KB (No ITCM Memory)

0101: 16 KB

0110: 32 KB

0111: 64 KB

Others: Reserved

- **DTCM_SIZE: Size of DTCM enabled memory block**

0000 0 KB (No DTCM Memory)

0101: 16 KB

0110: 32 KB

0111: 64 KB

Others: Reserved

19.5.4 EBI Chip Select Assignment Register

Register Name: EBI_CSA

Access Type: Read/Write

Reset: 0x0000_0000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	EBI_DBPUC

7	6	5	4	3	2	1	0
–	–	EBI_CS5A	EBI_CS4A	EBI_CS3A	–	EBI_CS1A	–

- **EBI_CS1A: EBI Chip Select 1 Assignment**

0 = EBI Chip Select 1 is assigned to the Static Memory Controller.

1 = EBI Chip Select 1 is assigned to the SDRAM Controller.

- **EBI_CS3A: EBI Chip Select 3 Assignment**

0 = EBI Chip Select 3 is only assigned to the Static Memory Controller and EBI_NCS3 behaves as defined by the SMC.

1 = EBI Chip Select 3 is assigned to the Static Memory Controller and the SmartMedia Logic is activated.

- **EBI_CS4A: EBI Chip Select 4 Assignment**

0 = EBI Chip Select 4 is only assigned to the Static Memory Controller and EBI_NCS4 behaves as defined by the SMC.

1 = EBI Chip Select 4 is assigned to the Static Memory Controller and the CompactFlash Logic (first slot) is activated.

- **EBI_CS5A: EBI Chip Select 5 Assignment**

0 = EBI Chip Select 5 is only assigned to the Static Memory Controller and EBI_NCS5 behaves as defined by the SMC.

1 = EBI Chip Select 5 is assigned to the Static Memory Controller and the CompactFlash Logic (second slot) is activated.

- **EBI_DBPUC: EBI Data Bus Pull-Up Configuration**

0 = EBI D0 - D15 Data Bus bits are internally pulled-up to the VDDIOM0 power supply.

1 = EBI D0 - D15 Data Bus bits are not internally pulled-up.

19.5.5 USB Pad Pull-up Control Register

Register Name: USB_PUCR

Access Type: Read/Write

Reset: 0x0000_0000

31	30	29	28	27	26	25	24
Reserved	UDP_PUP_ON	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **UDP_PUP_ON: UDP Pad Pull-up Enable**

0: Pad pull-up disabled

1: Pad pull-up enabled

20. External Bus Interface (EBI)

20.1 Overview

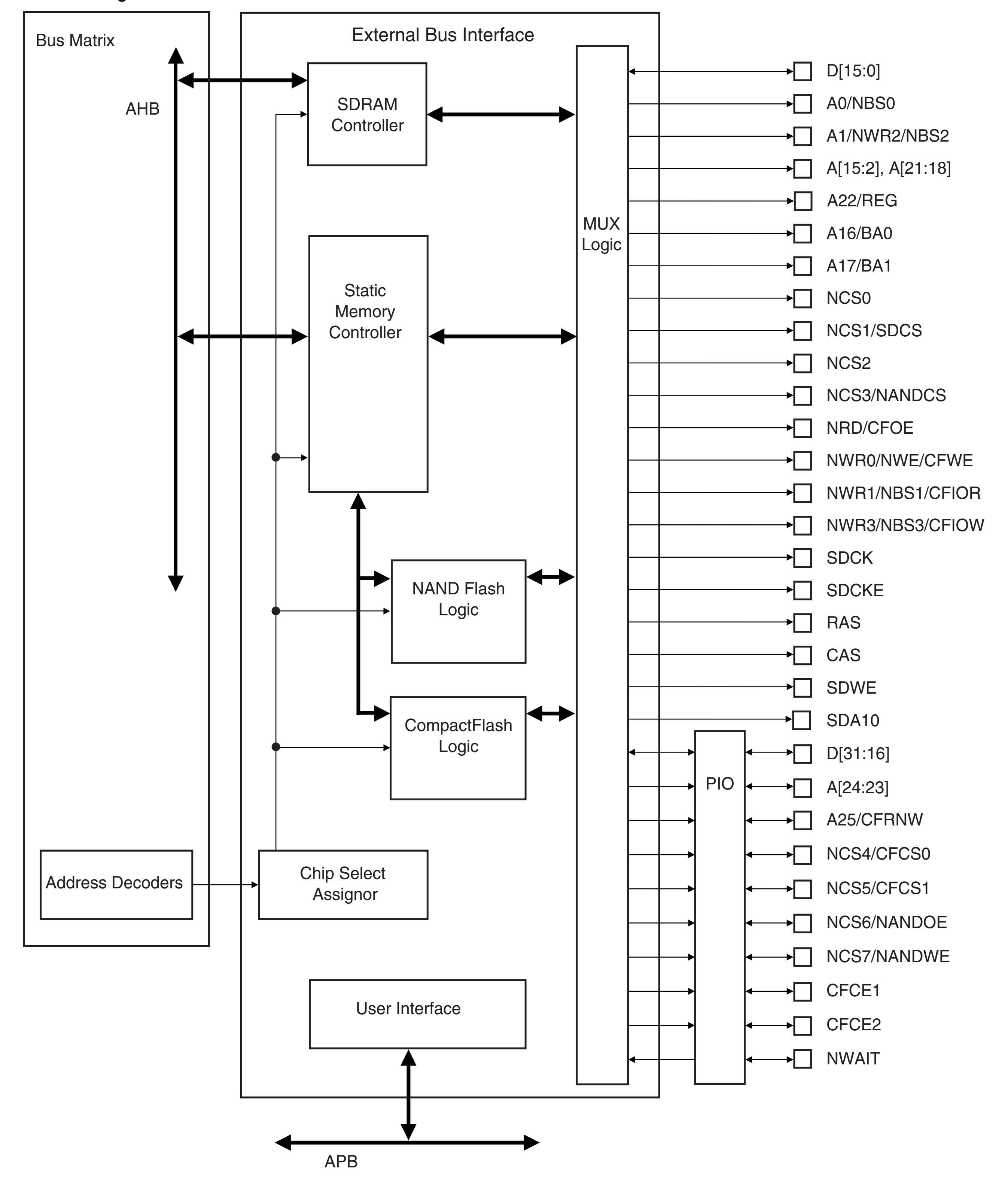
The External Bus Interface (EBI) is designed to ensure the successful data transfer between several external devices and the embedded Memory Controller of an ARM®-based device. The Static Memory and SDRAM Controllers are all featured external Memory Controllers on the EBI. These external Memory Controllers are capable of handling several types of external memory and peripheral devices, such as SRAM, PROM, EPROM, EEPROM, Flash, and SDRAM.

The EBI also supports the CompactFlash and the NAND Flash protocols via integrated circuitry that greatly reduces the requirements for external components. Furthermore, the EBI handles data transfers with up to eight external devices, each assigned to eight address spaces defined by the embedded Memory Controller. Data transfers are performed through a 16-bit or 32-bit data bus, an address bus of up to 26 bits, up to eight chip select lines (NCS[7:0]) and several control pins that are generally multiplexed between the different external Memory Controllers.

20.2 Block Diagram

Figure 20-1 shows the organization of the External Bus Interface.

Figure 20-1. Organization of the External Bus Interface



20.3 I/O Lines Description

Table 20-1. I/O Lines Description

Name	Function	Type	Active Level
EBI			
D0 - D31	Data Bus	I/O	
A0 - A25	Address Bus	Output	
NWAIT	External Wait Signal	Input	Low
SMC			
NCS0 - NCS7	Chip Select Lines	Output	Low
NWR0 - NWR3	Write Signals	Output	Low
NRD	Read Signal	Output	Low
NWE	Write Enable	Output	Low
NBS0 - NBS3	Byte Mask Signals	Output	Low
EBI for CompactFlash Support			
CFCE1 - CFCE2	CompactFlash Chip Enable	Output	Low
CFOE	CompactFlash Output Enable	Output	Low
CFWE	CompactFlash Write Enable	Output	Low
CFIOR	CompactFlash I/O Read Signal	Output	Low
CFIOW	CompactFlash I/O Write Signal	Output	Low
CFRNW	CompactFlash Read Not Write Signal	Output	
CFCS0 - CFCS1	CompactFlash Chip Select Lines	Output	Low
EBI for NAND Flash Support			
NANDCS	NAND Flash Chip Select Line	Output	Low
NANDOE	NAND Flash Output Enable	Output	Low
NANDWE	NAND Flash Write Enable	Output	Low
SDRAM Controller			
SDCK	SDRAM Clock	Output	
SDCKE	SDRAM Clock Enable	Output	High
SDCS	SDRAM Controller Chip Select Line	Output	Low
BA0 - BA1	Bank Select	Output	
SDWE	SDRAM Write Enable	Output	Low
RAS - CAS	Row and Column Signal	Output	Low
NWR0 - NWR3	Write Signals	Output	Low
NBS0 - NBS3	Byte Mask Signals	Output	Low
SDA10	SDRAM Address 10 Line	Output	

The connection of some signals through the MUX logic is not direct and depends on the Memory Controller in use at the moment.

Table 20-2 on page 133 details the connections between the two Memory Controllers and the EBI pins.

Table 20-2. EBI Pins and Memory Controllers I/O Lines Connections

EBI Pins	SDRAMC I/O Lines	SMC I/O Lines
NWR1/NBS1/CFIOR	NBS1	NWR1/NUB
A0/NBS0	Not Supported	SMC_A0/NLB
A1/NBS2/NWR2	Not Supported	SMC_A1
A[11:2]	SDRAMC_A[9:0]	SMC_A[11:2]
SDA10	SDRAMC_A10	Not Supported
A12	Not Supported	SMC_A12
A[14:13]	SDRAMC_A[12:11]	SMC_A[14:13]
A[25:15]	Not Supported	SMC_A[25:15]
D[31:16]	D[31:16]	D[31:16]
D[15:0]	D[15:0]	D[15:0]

20.4 Application Example

20.4.1 Hardware Interface

Table 20-3 and Table 20-4 detail the connections to be applied between the EBI pins and the external devices for each Memory Controller.

Table 20-3. EBI Pins and External Static Devices Connections

Pins	Pins of the Interfaced Device					
	8-bit Static Device	2 x 8-bit Static Devices	16-bit Static Device	4 x 8-bit Static Devices	2 x 16-bit Static Devices	32-bit Static Device
Controller	**SMC**					
D0 - D7	D0 - D7	D0 - D7	D0 - D7	D0 - D7	D0 - D7	D0 - D7
D8 - D15	–	D8 - D15	D8 - D15	D8 - D15	D8 - 15	D8 - 15
D16 - D23	–	–	–	D16 - D23	D16 - D23	D16 - D23
D24 - D31	–	–	–	D24 - D31	D24 - D31	D24 - D31
A0/NBS0	A0	–	NLB	–	NLB(3)	BE0(5)
A1/NWR2/NBS2	A1	A0	A0	WE(2)	NLB(4)	BE2(5)
A2 - A25	A[2:25]	A[1:24]	A[1:24]	A[0:23]	A[0:23]	A[0:23]
NCS0	CS	CS	CS	CS	CS	CS
NCS1/SDCS	CS	CS	CS	CS	CS	CS
NCS2	CS	CS	CS	CS	CS	CS
NCS3/NANDCS	CS	CS	CS	CS	CS	CS
NCS4/CFCS0	CS	CS	CS	CS	CS	CS



Table 20-3. EBI Pins and External Static Devices Connections (Continued)

Pins	Pins of the Interfaced Device					
	8-bit Static Device	2 x 8-bit Static Devices	16-bit Static Device	4 x 8-bit Static Devices	2 x 16-bit Static Devices	32-bit Static Device
Controller	**SMC**					
NCS5/CFCS1	CS	CS	CS	CS	CS	CS
NCS6/NAND0E	CS	CS	CS	CS	CS	CS
NCS7/NANDWE	CS	CS	CS	CS	CS	CS
NRD/CFOE	OE	OE	OE	OE	OE	OE
NWR0/NWE	WE	WE(1)	WE	WE(2)	WE	WE
NWR1/NBS1	–	WE(1)	NUB	WE(2)	NUB(3)	BE1(5)
NWR3/NBS3	–	–	–	WE(2)	NUB(4)	BE3(5)

Notes:
1. NWR1 enables upper byte writes. NWR0 enables lower byte writes.
2. NWRx enables corresponding byte x writes. (x = 0, 1, 2 or 3)
3. NBS0 and NBS1 enable respectively lower and upper bytes of the lower 16-bit word.
4. NBS2 and NBS3 enable respectively lower and upper bytes of the upper 16-bit word.
5. BEx: Byte x Enable (x = 0,1,2 or 3)

Table 20-4. EBI Pins and External Devices Connections

Pins	Pins of the Interfaced Device			
	SDRAM	Compact Flash	Compact Flash True IDE Mode	NAND Flash
Controller	**SDRAMC**	**SMC**		
D0 - D7	D0 - D7	D0 - D7	D0 - D7	AD0-AD7
D8 - D15	D8 - D15	D8 - 15	D8 - 15	AD8-AD15
D16 - D31	D16 - D31	–	–	–
A0/NBS0	DQM0	A0	A0	–
A1/NWR2/NBS2	DQM2	A1	A1	–
A2 - A10	A[0:8]	A[2:10]	A[2:10]	–
A11	A9	–	–	–
SDA10	A10	–	–	–
A12	–	–	–	–
A13 - A14	A[11:12]	–	–	–
A15	–	–	–	–
A16/BA0	BA0	–	–	–
A17/BA1	BA1	–	–	–
A18 - A20	–	–	–	–
A21	–	–	–	CLE(3)
A22	–	REG	REG	ALE(3)
A23 - A24	–	–	–	–

Table 20-4. EBI Pins and External Devices Connections (Continued)

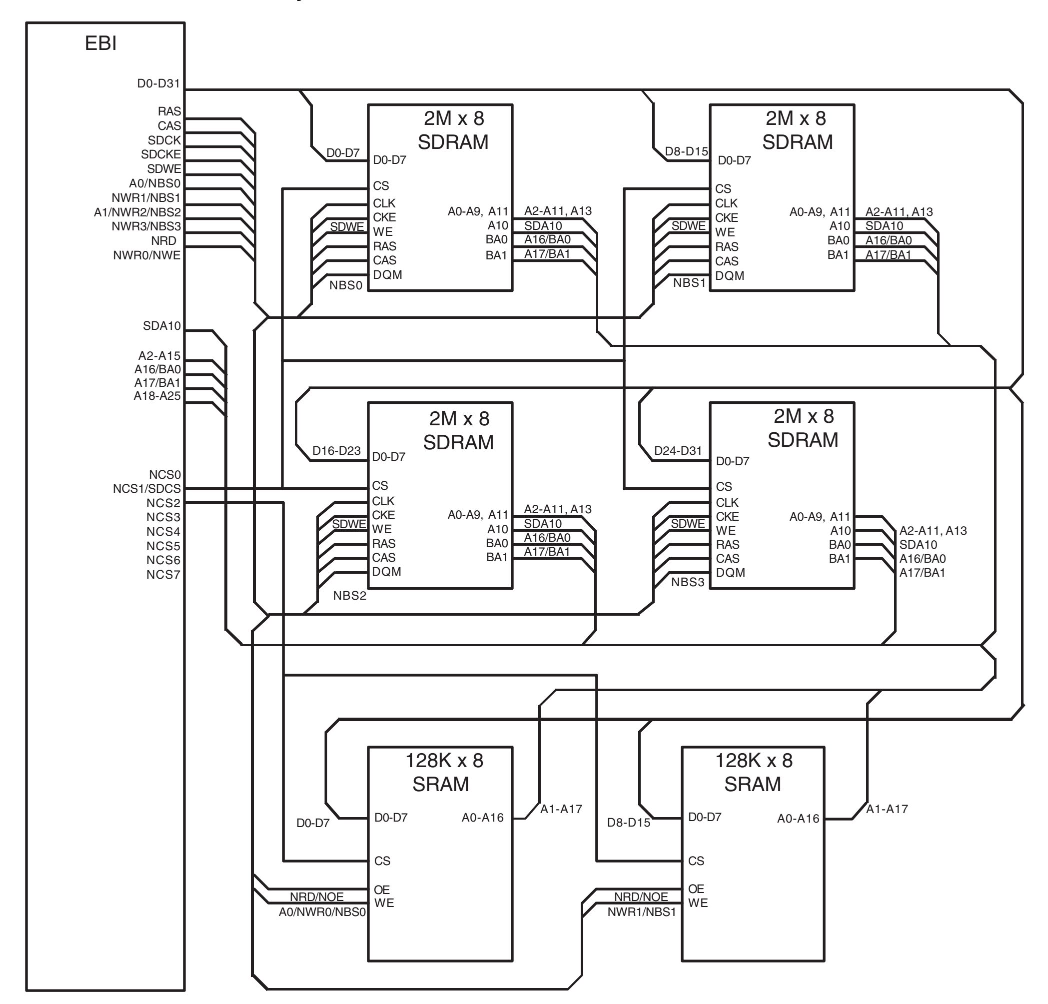
	Pins of the Interfaced Device			
Pins	SDRAM	Compact Flash	Compact Flash True IDE Mode	NAND Flash
Controller	**SDRAMC**	**SMC**		
A25	–	CFRNW(1)	CFRNW(1)	–
NCS0	–	–	–	–
NCS1/SDCS	CS	–	–	–
NCS2	–	–	–	–
NCS3/NANDCS	–	–	–	–
NCS4/CFCS0	–	CFCS0(1)	CFCS0(1)	–
NCS5/CFCS1	–	CFCS1(1)	CFCS1(1)	–
NCS6/NANDOE	–	–	–	OE
NCS7/NANDWE	–	–	–	WE
NRD/CFOE	–	OE	–	–
NWR0/NWE/CFWE	–	WE	WE	–
NWR1/NBS1/CFIOR	DQM1	IOR	IOR	–
NWR3/NBS3/CFIOW	DQM3	IOW	IOW	–
CFCE1	–	CE1	CS0	–
CFCE2	–	CE2	CS1	–
SDCK	CLK	–	–	–
SDCKE	CKE	–	–	–
RAS	RAS	–	–	–
CAS	CAS	–	–	–
SDWE	WE	–	–	–
NWAIT	–	WAIT	WAIT	–
Pxx(2)	–	CD1 or CD2	CD1 or CD2	–
Pxx(2)	–	–	–	CE
Pxx(2)	–	–	–	RDY

Note: 1. Not directly connected to the CompactFlash slot. Permits the control of the bidirectional buffer between the EBI data bus and the CompactFlash slot.
2. Any PIO line.
3. The CLE and ALE signals of the NAND Flash device may be driven by any address bit. For details, see "NAND Flash Support" on page 142.

20.4.2 Connection Examples

Figure 20-2 shows an example of connections between the EBI and external devices.

Figure 20-2. EBI Connections to Memory Devices



20.5 Product Dependencies

20.5.1 I/O Lines

The pins used for interfacing the External Bus Interface may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the External Bus Interface pins to their peripheral function. If I/O lines of the External Bus Interface are not used by the application, they can be used for other purposes by the PIO Controller.

20.6 Functional Description

The EBI transfers data between the internal AHB Bus (handled by the Bus Matrix) and the external memories or peripheral devices. It controls the waveforms and the parameters of the external address, data and control busses and is composed of the following elements:

- Static Memory Controller (SMC)
- SDRAM Controller (SDRAMC)
- A chip select assignment feature that assigns an AHB address space to the external devices
- A multiplex controller circuit that shares the pins between the different Memory Controllers
- Programmable CompactFlash support logic
- Programmable NAND Flash support logic

20.6.1 Bus Multiplexing

The EBI offers a complete set of control signals that share the 32-bit data lines, the address lines of up to 26 bits and the control signals through a multiplex logic operating in function of the memory area requests.

Multiplexing is specifically organized in order to guarantee the maintenance of the address and output control lines at a stable state while no external access is being performed. Multiplexing is also designed to respect the data float times defined in the Memory Controllers. Furthermore, refresh cycles of the SDRAM are executed independently by the SDRAM Controller without delaying the other external Memory Controller accesses.

20.6.2 Pull-up Control

The EBI_CSA register in the Bus Matrix User Interface permits enabling of on-chip pull-up resistors on the data bus lines not multiplexed with the PIO Controller lines. The pull-up resistors are enabled after reset. Setting the DBPUC bit disables the pull-up resistors on the D0 to D15 lines. Enabling the pull-up resistor on the D16-D31 lines can be performed by programming the appropriate PIO controller.

20.6.3 Static Memory Controller

For information on the Static Memory Controller, refer to the Static Memory Controller section.

20.6.4 SDRAM Controller

For information on the SDRAM Controller, refer to the SDRAM section.

20.6.5 CompactFlash Support

The External Bus Interface integrates circuitry that interfaces to CompactFlash devices.

The CompactFlash logic is driven by the Static Memory Controller (SMC) on the NCS4 and/or NCS5 address space. Programming the CS4A and/or CS5A bit of the EBI_CSA Register to the appropriate value enables this logic. For details on this register, refer to the Bus Matrix User Interface section. Access to an external CompactFlash device is then made by accessing the address space reserved to NCS4 and/or NCS5 (i.e., between 0x5000 0000 and 0x5FFF FFFF for NCS4 and between 0x6000 0000 and 0x6FFF FFFF for NCS5).

All CompactFlash modes (Attribute Memory, Common Memory, I/O and True IDE) are supported but the signals _IOIS16 (I/O and True IDE modes) and _ATA SEL (True IDE mode) are not handled.

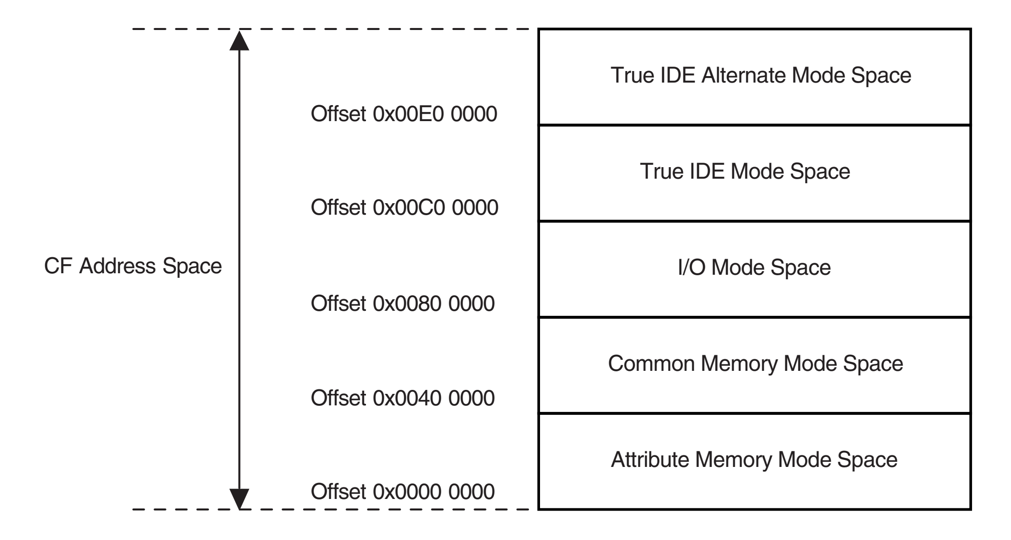


20.6.5.1 I/O Mode, Common Memory Mode, Attribute Memory Mode and True IDE Mode

Within the NCS4 and/or NCS5 address space, the current transfer address is used to distinguish I/O mode, common memory mode, attribute memory mode and True IDE mode.

The different modes are accessed through a specific memory mapping as illustrated on Figure 20-3. A[23:21] bits of the transfer address are used to select the desired mode as described in Table 20-5 on page 138.

Figure 20-3. CompactFlash Memory Mapping



Note: The A22 pin of the EBI is used to drive the REG signal of the CompactFlash Device (except in True IDE mode).

Table 20-5. CompactFlash Mode Selection

A[23:21]	Mode Base Address
000	Attribute Memory
010	Common Memory
100	I/O Mode
110	True IDE Mode
111	Alternate True IDE Mode

20.6.5.2 CFCE1 and CFCE2 signals

To cover all types of access, the SMC must be alternatively set to drive 8-bit data bus or 16-bit data bus. The odd byte access on the D[7:0] bus is only possible when the SMC is configured to drive 8-bit memory devices on the corresponding NCS pin (NCS4 and or NCS5). The Chip Select Register (DBW field in the corresponding Chip Select Mode Register) of the NCS4 and/or NCS5 address space must be set as shown in Table 20-6 to enable the required access type.

NBS1 and NBS0 are the byte selection signals from SMC and are available when the SMC is set in Byte Select mode on the corresponding Chip Select.

The CFCE1 and CFCE2 waveforms are identical to the corresponding NCSx waveform. For details on these waveforms and timings, refer to the Static Memory Controller section.

Table 20-6. CFCE1 and CFCE2 Truth Table

Mode	CFCE2	CFCE1	DBW	Comment	SMC Access Mode
Attribute Memory	NBS1	NBS0	16 bits	Access to Even Byte on D[7:0]	Byte Select
Common Memory	NBS1	NBS0	16bits	Access to Even Byte on D[7:0] Access to Odd Byte on D[15:8]	Byte Select
	1	0	8 bits	Access to Odd Byte on D[7:0]	Don't Care
I/O Mode	NBS1	NBS0	16 bits	Access to Even Byte on D[7:0] Access to Odd Byte on D[15:8]	Byte Select
	1	0	8 bits	Access to Odd Byte on D[7:0]	Don't Care
True IDE Mode					
Task File	1	0	8 bits	Access to Even Byte on D[7:0] Access to Odd Byte on D[7:0]	Don't Care
Data Register	1	0	16 bits	Access to Even Byte on D[7:0] Access to Odd Byte on D[15:8]	Byte Select
Alternate True IDE Mode					
Control Register Alternate Status Read	0	1	Don't Care	Access to Even Byte on D[7:0]	Don't Care
Drive Address	0	1	8 bits	Access to Odd Byte on D[7:0]	Don't Care
True IDE Standby Mode or Address Space is not assigned to CF	1	1	Don't Care	Don't Care	Don't Care

20.6.5.3 Read/Write Signals

In I/O mode and True IDE mode, the CompactFlash logic drives the read and write command signals of the SMC on CFIOR and CFIOW signals, while the CFOE and CFWE signals are deactivated. Likewise, in common memory mode and attribute memory mode, the SMC signals are driven on the CFOE and CFWE signals, while the CFIOR and CFIOW are deactivated. Figure 20-4 on page 140 demonstrates a schematic representation of this logic.

Attribute memory mode, common memory mode and I/O mode are supported by setting the address setup and hold time on the NCS4 (and/or NCS5) chip select to the appropriate values.

Figure 20-4. CompactFlash Read/Write Control Signals

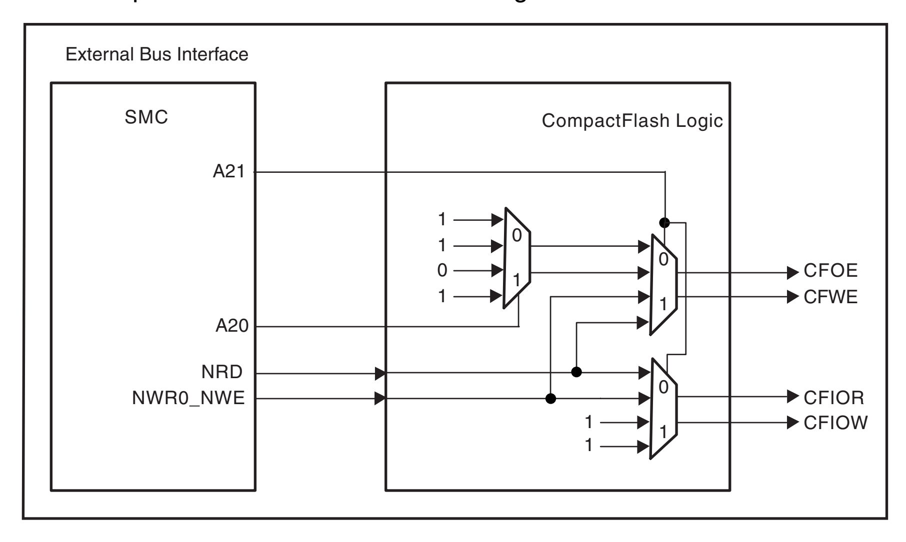


Table 20-7. CompactFlash Mode Selection

Mode Base Address	CFOE	CFWE	CFIOR	CFIOW
Attribute Memory Common Memory	NRD	NWR0_NWE	1	1
I/O Mode	1	1	NRD	NWR0_NWE
True IDE Mode	0	1	NRD	NWR0_NWE

20.6.5.4 Multiplexing of CompactFlash Signals on EBI Pins

Table 20-8 on page 140 and Table 20-9 on page 141 illustrate the multiplexing of the CompactFlash logic signals with other EBI signals on the EBI pins. The EBI pins in Table 20-8 are strictly dedicated to the CompactFlash interface as soon as the CS4A and/or CS5A field of the EBI_CSA Register is set. These pins must not be used to drive any other memory devices.

The EBI pins in Table 20-9 on page 141 remain shared between all memory areas when the corresponding CompactFlash interface is enabled (CS4A = 1 and/or CS5A = 1).

Table 20-8. Dedicated CompactFlash Interface Multiplexing

Pins	CompactFlash Signals		EBI Signals	
	CS4A = 1	**CS5A = 1**	**CS4A = 0**	**CS5A = 0**
NCS4/CFCS0	CFCS0		NCS4	
NCS5/CFCS1		CFCS1		NCS5

Table 20-9. Shared CompactFlash Interface Multiplexing

Pins	Access to CompactFlash Device	Access to Other EBI Devices
	CompactFlash Signals	**EBI Signals**
NRD/CFOE	CFOE	NRD
NWR0/NWE/CFWE	CFWE	NWR0/NWE
NWR1/NBS1/CFIOR	CFIOR	NWR1/NBS1
NWR3/NBS3/CFIOW	CFIOW	NWR3/NBS3
A25/CFRNW	CFRNW	A25

20.6.5.5 Application Example

Figure 20-5 on page 142 illustrates an example of a CompactFlash application. CFCS0 and CFRNW signals are not directly connected to the CompactFlash slot 0, but do control the direction and the output enable of the buffers between the EBI and the CompactFlash Device. The timing of the CFCS0 signal is identical to the NCS4 signal. Moreover, the CFRNW signal remains valid throughout the transfer, as does the address bus. The CompactFlash _WAIT signal is connected to the NWAIT input of the Static Memory Controller. For details on these waveforms and timings, refer to the Static Memory Controller section.

Figure 20-5. CompactFlash Application Example

20.6.6 NAND Flash Support

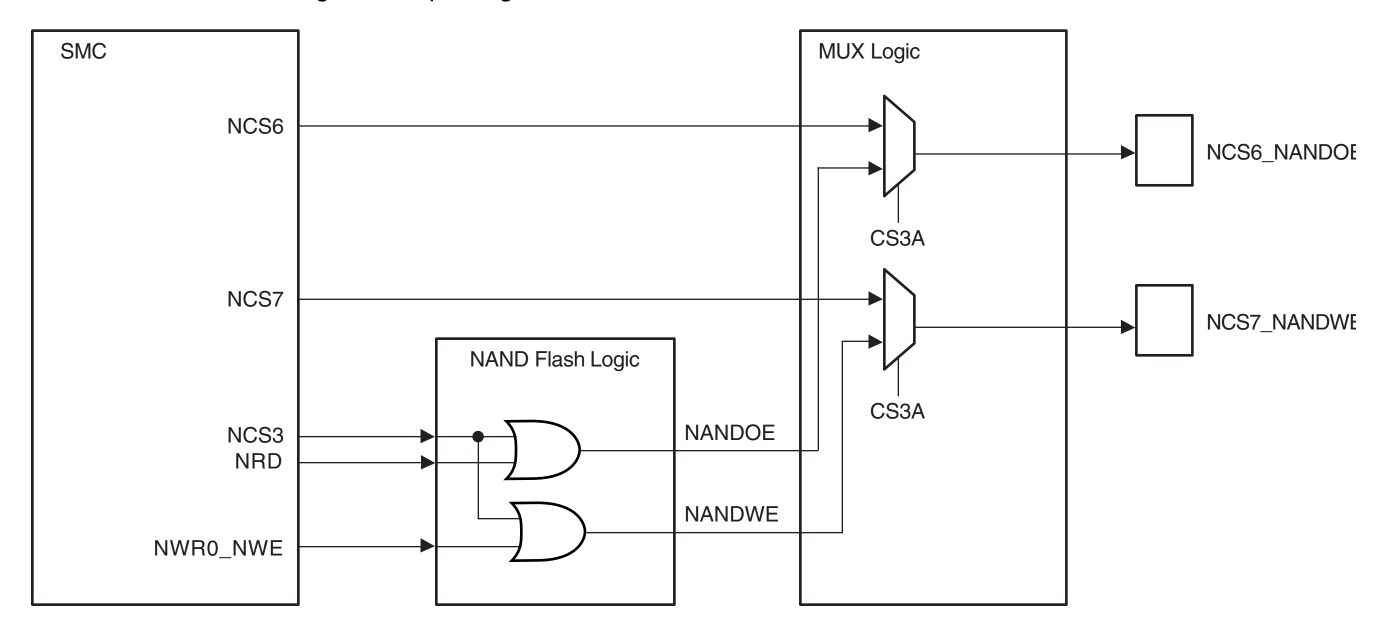
The EBI integrates circuitry that interfaces to NAND Flash devices.

The NAND Flash logic is driven by the Static Memory Controller on the NCS3 address space. Programming the CS3A field in the EBI_CSA Register in the Bus Matrix User Interface to the appropriate value enables the NAND Flash logic. For details on this register, refer to the Bus Matrix User Interface section. Access to an external NAND Flash device is then made by accessing the address space reserved to NCS3 (i.e., between 0x40000000 and 0x4FFF FFFF).

The NAND Flash Logic drives the read and write command signals of the SMC on the NANDOE and NANDWE signals when the NCS3 signal is active. NANDOE and NANDWE are invalidated as soon as the transfer address fails to lie in the NCS3 address space. For details on these waveforms, refer to the Static Memory Controller section.

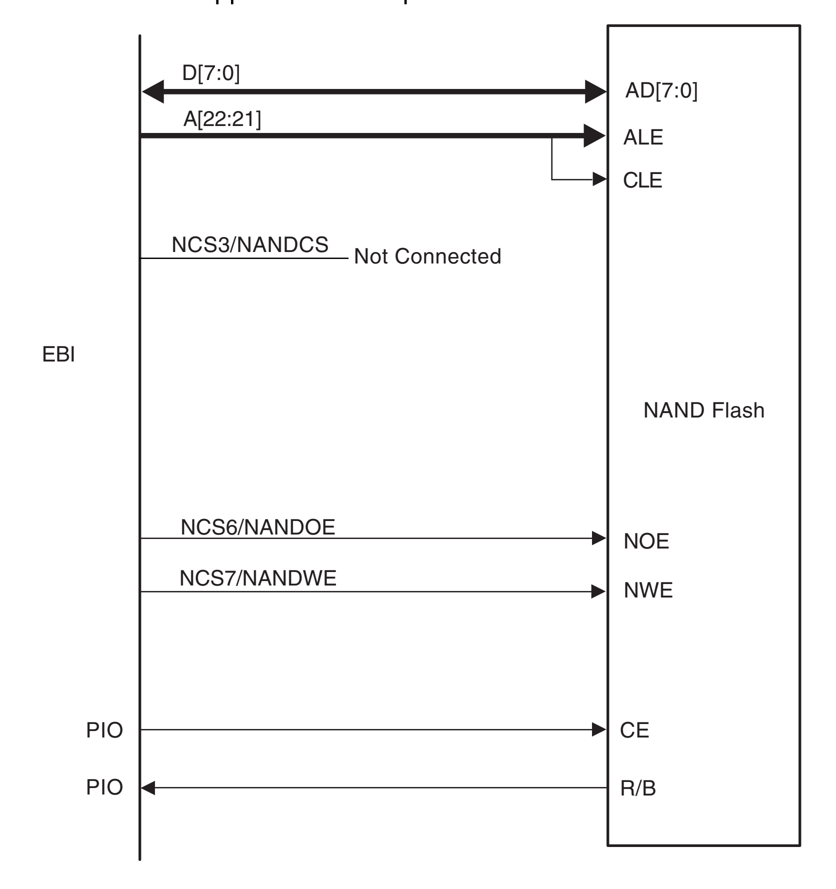
The NANDOE and NANDWE signals are multiplexed with NCS6 and NCS7 signals of the Static Memory Controller. This multiplexing is controlled in the MUX logic part of the EBI by the CS3A bit in the in the EBI_CSA Register For details on this register, refer to the Bus Matrix User Interface Section. NCS6 and NCS7 become unavailable. Performing an access within the address space reserved to NCS6 and NCS7 (i.e., between 0x70000000 and 0x8FFF FFFF) may lead to an unpredictable outcome.

Figure 20-6. NAND Flash Signal Multiplexing on EBI Pins



The address latch enable and command latch enable signals on the NAND Flash device are driven by address bits A22 and A21 of the EBI address bus. The user should note that any bit on the EBI address bus can also be used for this purpose. The command, address or data words on the data bus of the NAND Flash device are distinguished by using their address within the NCS3 address space. The chip enable (CE) signal of the device and the ready/busy (R/B) signals are connected to PIO lines. The CE signal then remains asserted even when NCS3 is not selected, preventing the device from returning to standby mode.

Figure 20-7. NAND Flash Application Example



Note: The External Bus Interface is also able to support 16-bits devices.

21. Static Memory Controller (SMC)

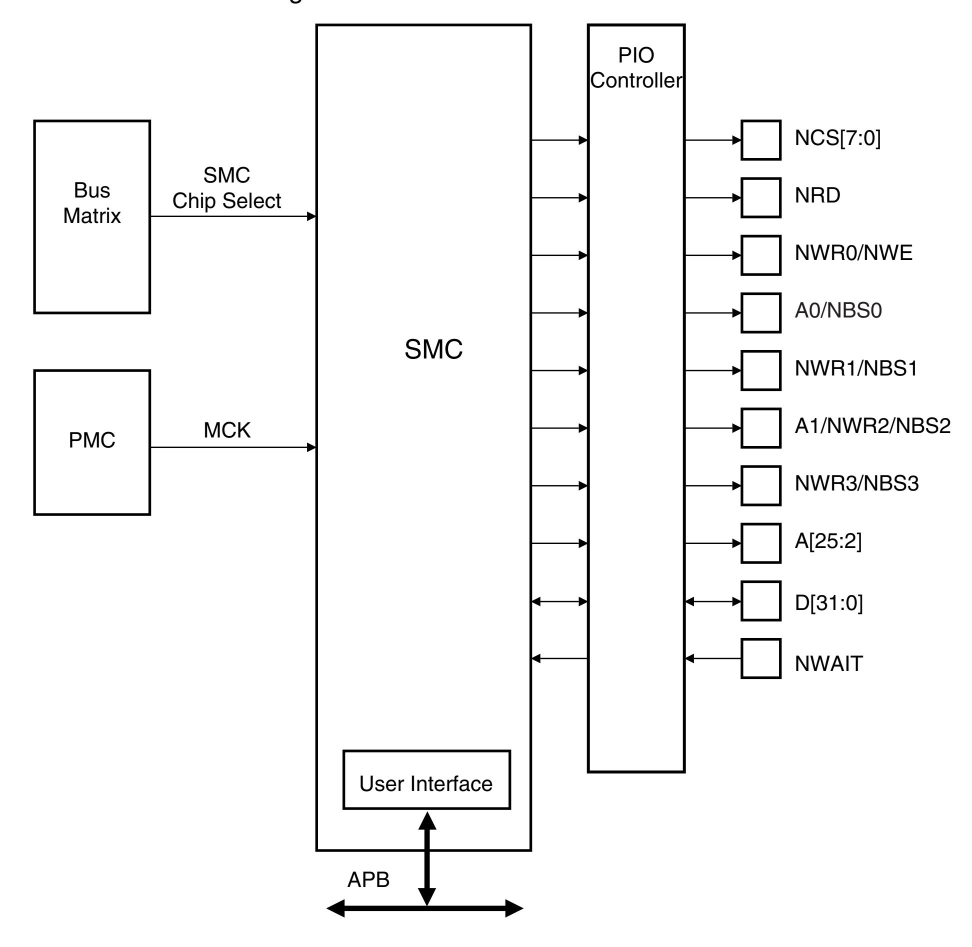
21.1 Overview

The Static Memory Controller (SMC) generates the signals that control the access to the external memory devices or peripheral devices. It has 8 Chip Selects and a 26-bit address bus. The 32-bit data bus can be configured to interface with 8-, 16-, or 32-bit external devices. Separate read and write control signals allow for direct memory and peripheral interfacing. Read and write signal waveforms are fully parametrizable.

The SMC can manage wait requests from external devices to extend the current access. The SMC is provided with an automatic slow clock mode. In slow clock mode, it switches from user-programmed waveforms to slow-rate specific waveforms on read and write signals. The SMC supports asynchronous burst read in page mode access for page size up to 32 bytes.

21.2 Block Diagram

Figure 21-1. SMC Block Diagram



21.3 I/O Lines Description

Table 21-1. I/O Line Description

Name	Description	Type	Active Level
NCS[7:0]	Static Memory Controller Chip Select Lines	Output	Low
NRD	Read Signal	Output	Low
NWR0/NWE	Write 0/Write Enable Signal	Output	Low
A0/NBS0	Address Bit 0/Byte 0 Select Signal	Output	Low
NWR1/NBS1	Write 1/Byte 1 Select Signal	Output	Low
A1/NWR2/NBS2	Address Bit 1/Write 2/Byte 2 Select Signal	Output	Low
NWR3/NBS3	Write 3/Byte 3 Select Signal	Output	Low
A[25:2]	Address Bus	Output	
D[31:0]	Data Bus	I/O	
NWAIT	External Wait Signal	Input	Low

21.4 Multiplexed Signals

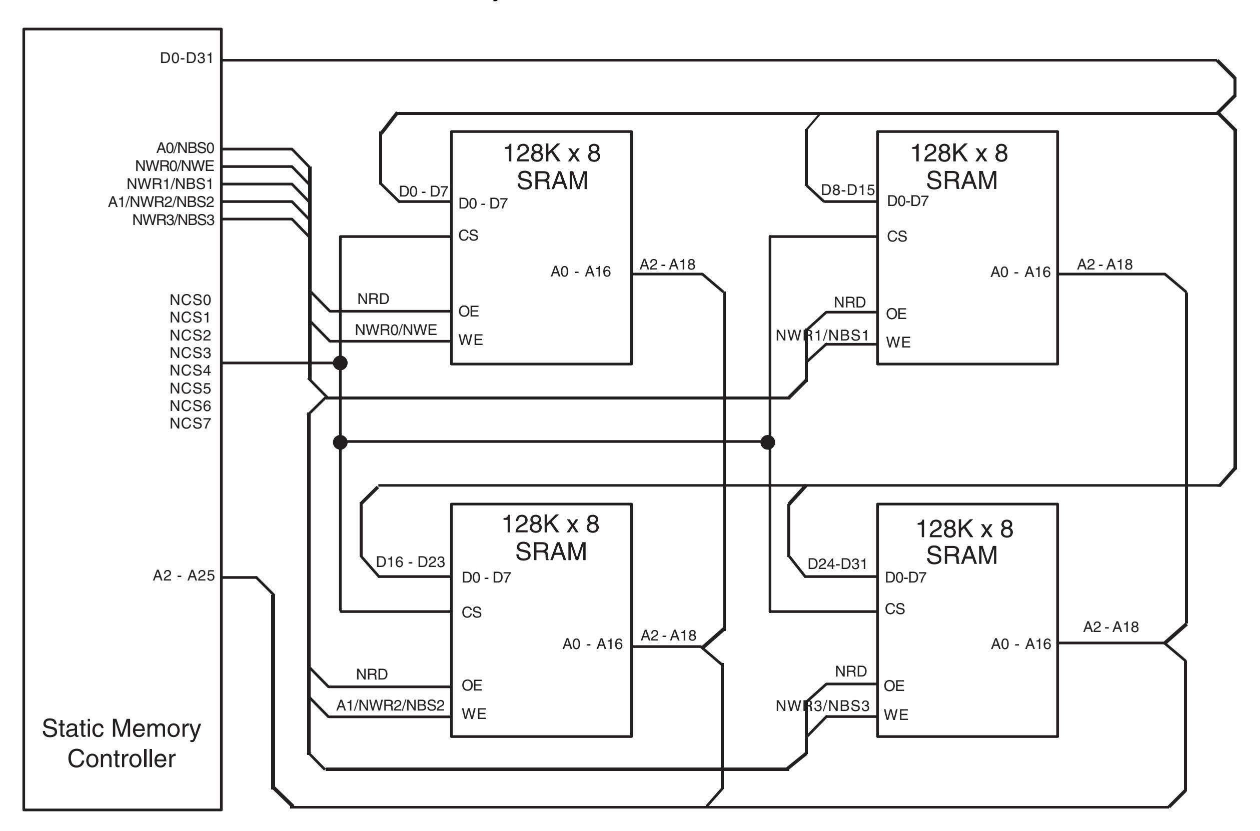
Table 21-2. Static Memory Controller (SMC) Multiplexed Signals

Multiplexed Signals			Related Function
NWR0	NWE		Byte-write or byte-select access, see ”Byte Write or Byte Select Access” on page 148
A0	NBS0		8-bit or 16-/32-bit data bus, see ”Data Bus Width” on page 148
NWR1	NBS1		Byte-write or byte-select access see ”Byte Write or Byte Select Access” on page 148
A1	NWR2	NBS2	8-/16-bit or 32-bit data bus, see ”Data Bus Width” on page 148. Byte-write or byte-select access, see ”Byte Write or Byte Select Access” on page 148
NWR3	NBS3		Byte-write or byte-select access see ”Byte Write or Byte Select Access” on page 148

21.5 Application Example

21.5.1 Hardware Interface

Figure 21-2. SMC Connections to Static Memory Devices



21.6 Product Dependencies

21.6.1 I/O Lines

The pins used for interfacing the Static Memory Controller may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the Static Memory Controller pins to their peripheral function. If I/O Lines of the SMC are not used by the application, they can be used for other putposes by the PIO Controller.



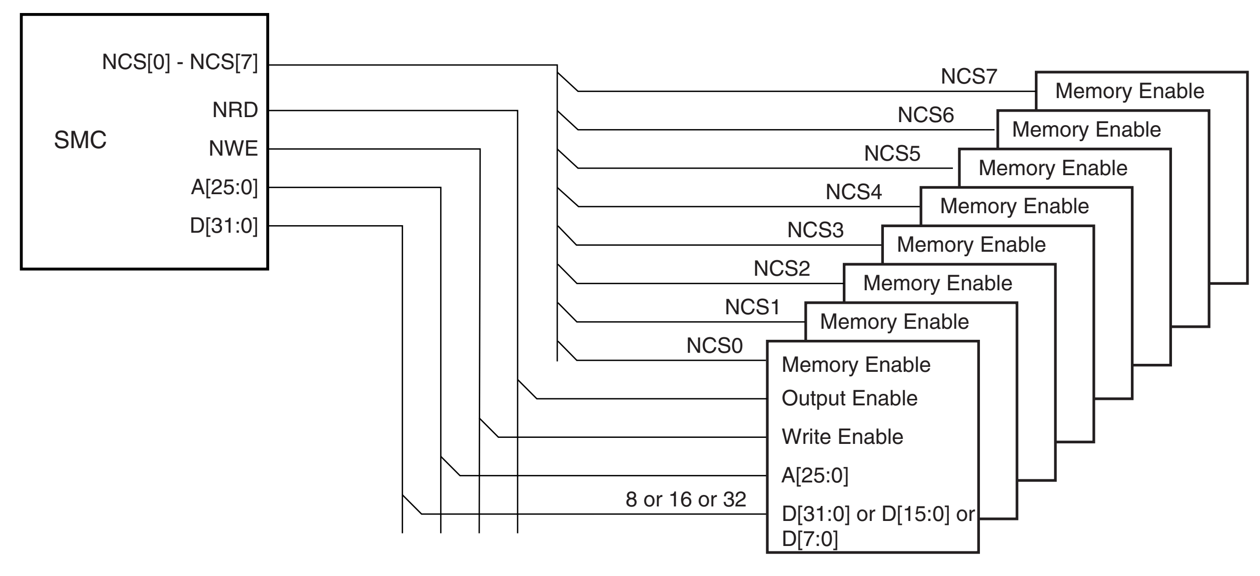
21.7 External Memory Mapping

The SMC provides up to 26 address lines, A[25:0]. This allows each chip select line to address up to 64 Mbytes of memory.

If the physical memory device connected on one chip select is smaller than 64 Mbytes, it wraps around and appears to be repeated within this space. The SMC correctly handles any valid access to the memory device within the page (see Figure 21-1).

A[25:0] is only significant for 8-bit memory, A[25:1] is used for 16-bit memory, A[25:2] is used for 32-bit memory.

Figure 21-3. Memory Connections for Eight External Devices



21.8 Connection to External Devices

21.8.1 Data Bus Width

A data bus width of 8, 16, or 32 bits can be selected for each chip select. This option is controlled by the field DBW in SMC_MODE (Mode Register) for the corresponding chip select.

Figure 21-4 shows how to connect a 512K x 8-bit memory on NCS2. Figure 21-5 shows how to connect a 512K x 16-bit memory on NCS2. Figure 21-6 shows two 16-bit memories connected as a single 32-bit memory

21.8.2 Byte Write or Byte Select Access

Each chip select with a 16-bit or 32-bit data bus can operate with one of two different types of write access: byte write or byte select access. This is controlled by the BAT field of the SMC_MODE register for the corresponding chip select.

Figure 21-4. Memory Connection for an 8-bit Data Bus

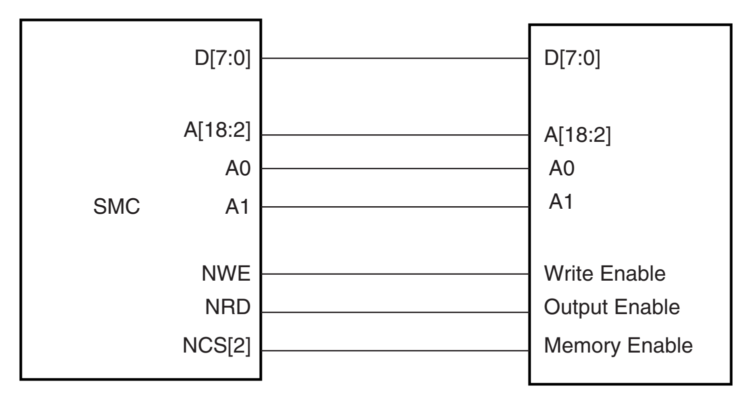


Figure 21-5. Memory Connection for a 16-bit Data Bus

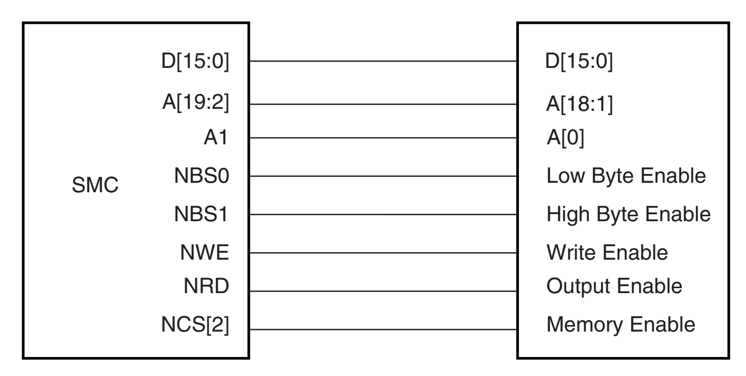
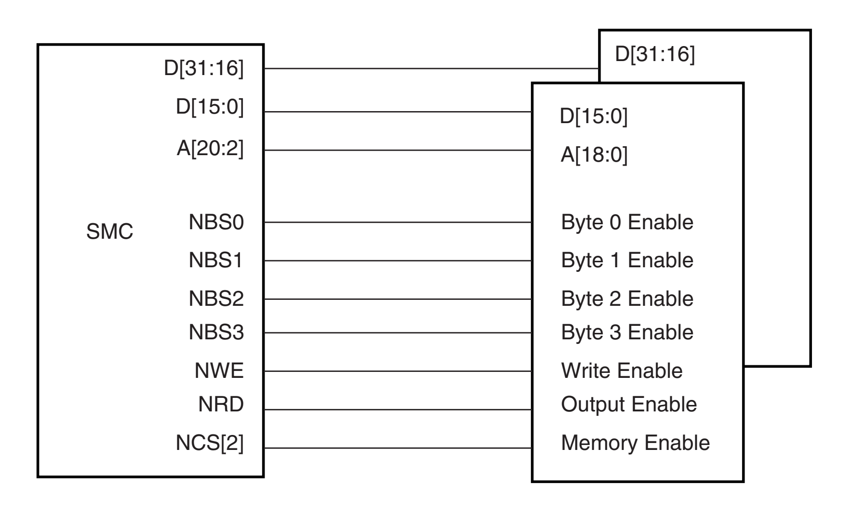


Figure 21-6. Memory Connection for a 32-bit Data Bus



21.8.2.1 Byte Write Access

Byte write access supports one byte write signal per byte of the data bus and a single read signal.

- For 16-bit devices: the SMC provides NWR0 and NWR1 write signals for respectively byte0 (lower byte) and byte1 (upper byte) of a 16-bit bus. One single read signal (NRD) is provided.

  Byte Write Access is used to connect 2 x 8-bit devices as a 16-bit memory.

- For 32-bit devices: NWR0, NWR1, NWR2 and NWR3, are the write signals of byte0 (lower byte), byte1, byte2 and byte 3 (upper byte) respectively. One single read signal (NRD) is provided.

  Byte Write Access is used to connect 4 x 8-bit devices as a 32-bit memory.

  Byte Write option is illustrated on Figure 21-7.

21.8.2.2 Byte Select Access

In this mode, read/write operations can be enabled/disabled at a byte level. One byte-select line per byte of the data bus is provided. One NRD and one NWE signal control read and write.

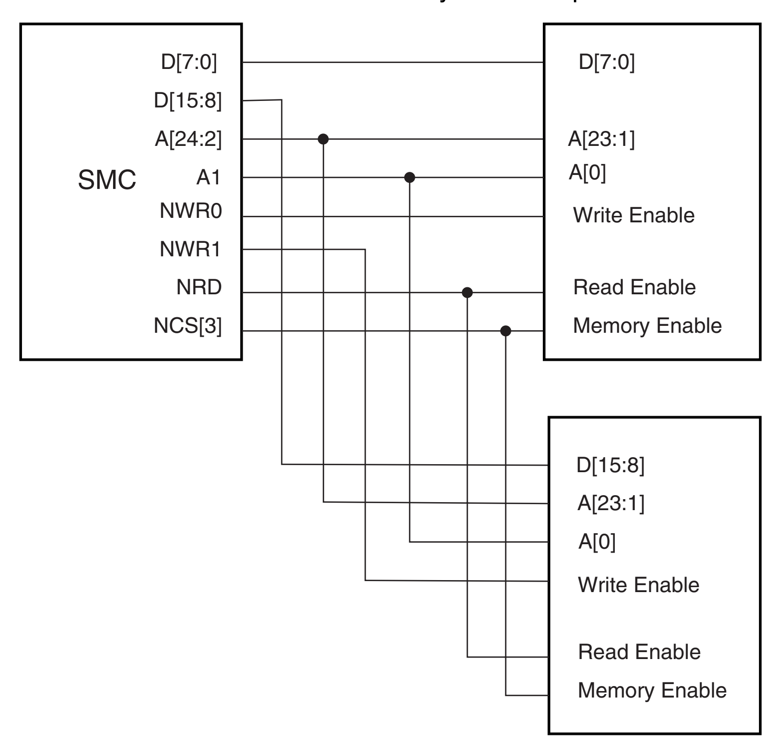
- For 16-bit devices: the SMC provides NBS0 and NBS1 selection signals for respectively byte0 (lower byte) and byte1 (upper byte) of a 16-bit bus.

  Byte Select Access is used to connect one 16-bit device.

- For 32-bit devices: NBS0, NBS1, NBS2 and NBS3, are the selection signals of byte0 (lower byte), byte1, byte2 and byte 3 (upper byte) respectively. Byte Select Access is used to connect two 16-bit devices.

Figure 21-8 shows how to connect two 16-bit devices on a 32-bit data bus in Byte Select Access mode, on NCS3 (BAT = Byte Select Access).

Figure 21-7. Connection of 2 x 8-bit Devices on a 16-bit Bus: Byte Write Option



21.8.2.3 Signal Multiplexing

Depending on the BAT, only the write signals or the byte select signals are used. To save IOs at the external bus interface, control signals at the SMC interface are multiplexed. Table 21-3 shows signal multiplexing depending on the data bus width and the byte access type.

For 32-bit devices, bits A0 and A1 are unused. For 16-bit devices, bit A0 of address is unused. When Byte Select Option is selected, NWR1 to NWR3 are unused. When Byte Write option is selected, NBS0 to NBS3 are unused.



Figure 21-8. Connection of 2x16-bit Data Bus on a 32-bit Data Bus (Byte Select Option)

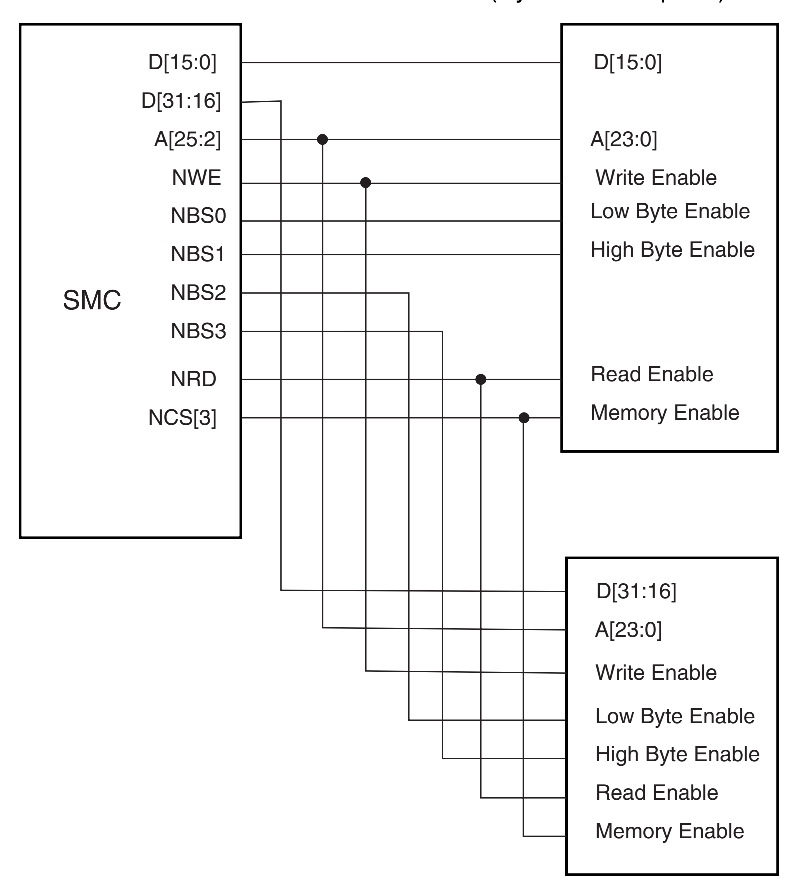


Table 21-3. SMC Multiplexed Signal Translation

Signal Name	**32-bit Bus**			**16-bit Bus**		**8-bit Bus**
Device Type	**1x32-bit**	**2x16-bit**	**4 x 8-bit**	**1x16-bit**	**2 x 8-bit**	**1 x 8-bit**
Byte Access Type (BAT)	Byte Select	Byte Select	Byte Write	Byte Select	Byte Write	
NBS0_A0	NBS0	NBS0		NBS0		A0
NWE_NWR0	NWE	NWE	NWR0	NWE	NWR0	NWE
NBS1_NWR1	NBS1	NBS1	NWR1	NBS1	NWR1	
NBS2_NWR2_A1	NBS2	NBS2	NWR2	A1	A1	A1
NBS3_NWR3	NBS3	NBS3	NWR3			

21.9 Standard Read and Write Protocols

In the following sections, the byte access type is not considered. Byte select lines (NBS0 to NBS3) always have the same timing as the A address bus. NWE represents either the NWE signal in byte select access type or one of the byte write lines (NWR0 to NWR3) in byte write access type. NWR0 to NWR3 have the same timings and protocol as NWE. In the same way, NCS represents one of the NCS[0..7] chip select lines.

21.9.1 Read Waveforms

The read cycle is shown on Figure 21-9.

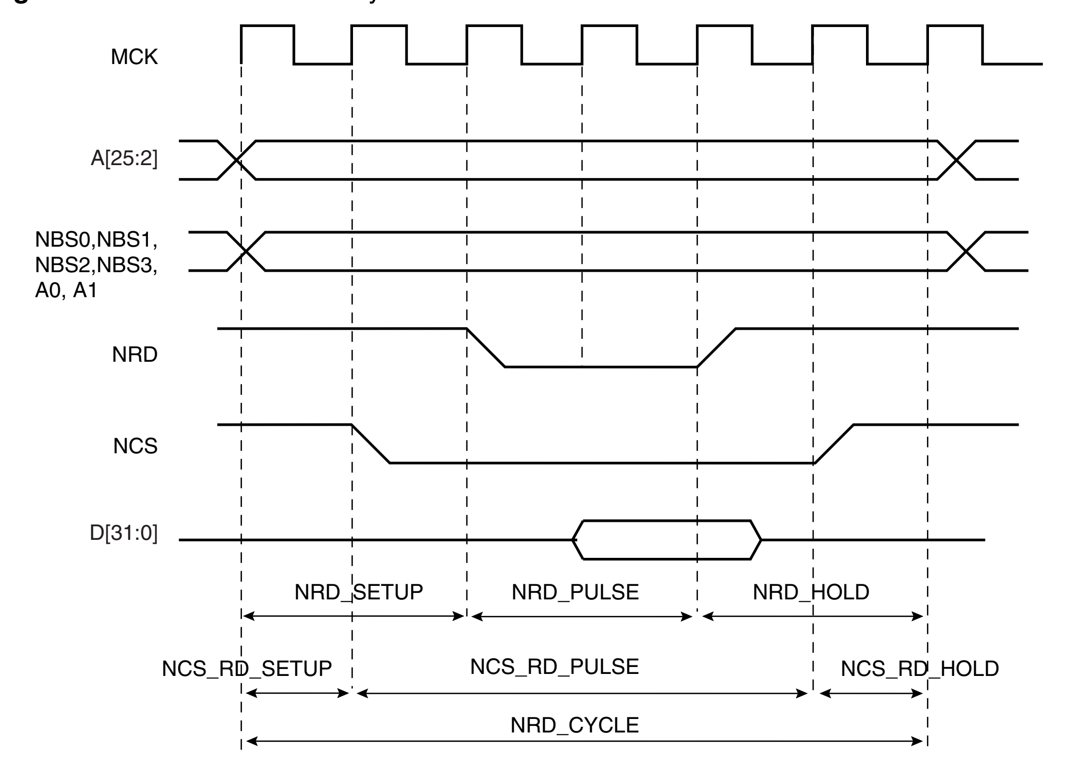
The read cycle starts with the address setting on the memory address bus, i.e.:

{A[25:2], A1, A0} for 8-bit devices

{A[25:2], A1} for 16-bit devices

A[25:2] for 32-bit devices.

Figure 21-9. Standard Read Cycle



21.9.1.1 NRD Waveform

The NRD signal is characterized by a setup timing, a pulse width and a hold timing.

a. NRD_SETUP: the NRD setup time is defined as the setup of address before the NRD falling edge;

b. NRD_PULSE: the NRD pulse length is the time between NRD falling edge and NRD rising edge;

c. NRD_HOLD: the NRD hold time is defined as the hold time of address after the NRD rising edge.

21.9.1.2 NCS Waveform

Similarly, the NCS signal can be divided into a setup time, pulse length and hold time:

a. NCS_RD_SETUP: the NCS setup time is defined as the setup time of address before the NCS falling edge.
b. NCS_RD_PULSE: the NCS pulse length is the time between NCS falling edge and NCS rising edge;
c. NCS_RD_HOLD: the NCS hold time is defined as the hold time of address after the NCS rising edge.

21.9.1.3 Read Cycle

The NRD_CYCLE time is defined as the total duration of the read cycle, i.e., from the time where address is set on the address bus to the point where address may change. The total read cycle time is equal to:

NRD_CYCLE = NRD_SETUP + NRD_PULSE + NRD_HOLD

= NCS_RD_SETUP + NCS_RD_PULSE + NCS_RD_HOLD

All NRD and NCS timings are defined separately for each chip select as an integer number of Master Clock cycles. To ensure that the NRD and NCS timings are coherent, user must define the total read cycle instead of the hold timing. NRD_CYCLE implicitly defines the NRD hold time and NCS hold time as:

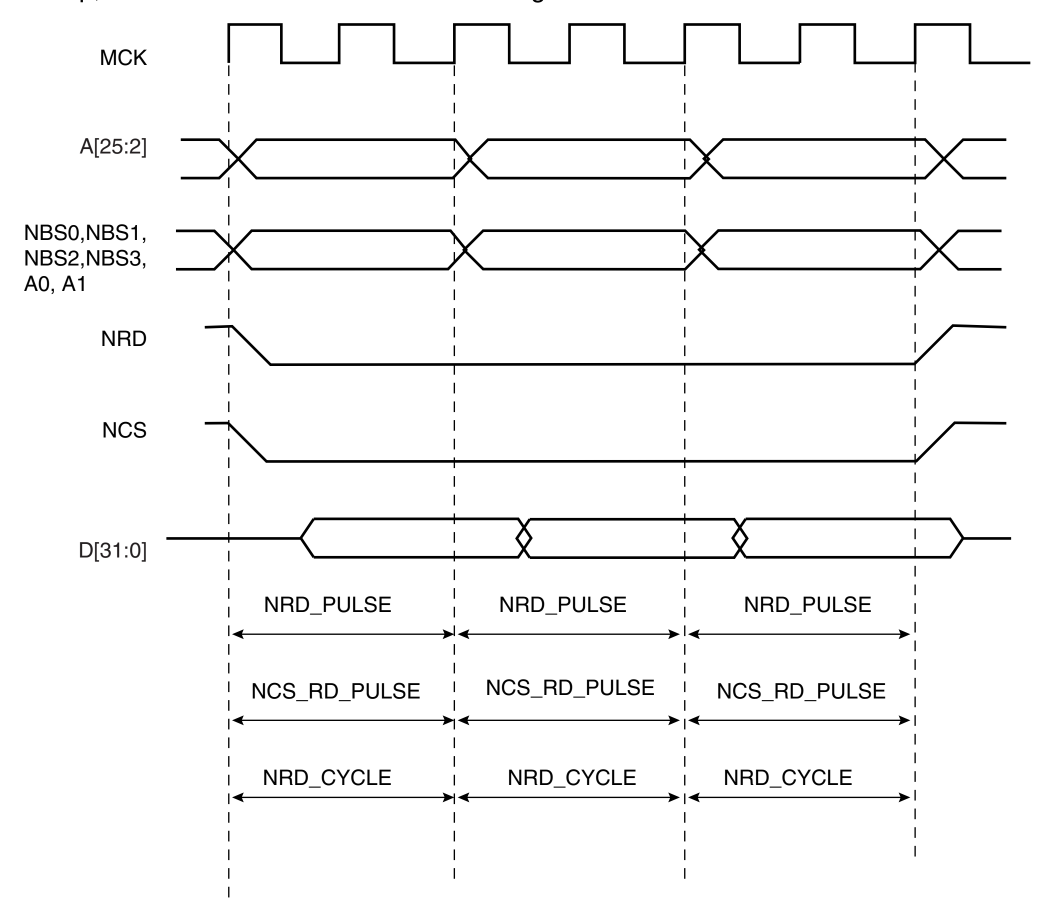
NRD_HOLD = NRD_CYCLE - NRD SETUP - NRD PULSE

NCS_RD_HOLD = NRD_CYCLE - NCS_RD_SETUP - NCS_RD_PULSE

21.9.1.4 Null Delay Setup and Hold

If null setup and hold parameters are programmed for NRD and/or NCS, NRD and NCS remain active continuously in case of consecutive read cycles in the same memory (see Figure 21-10).

Figure 21-10. No Setup, No Hold On NRD and NCS Read Signals



21.9.1.5 Null Pulse

Programming null pulse is not permitted. Pulse must be at least set to 1. A null value leads to unpredictable behavior.

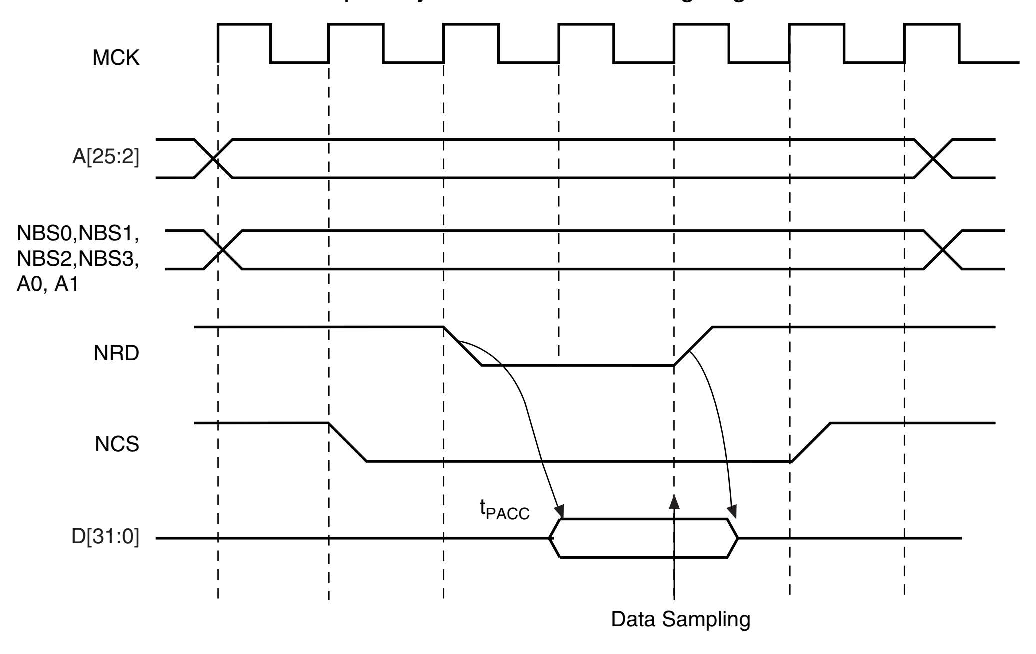
21.9.2 Read Mode

As NCS and NRD waveforms are defined independently of one other, the SMC needs to know when the read data is available on the data bus. The SMC does not compare NCS and NRD timings to know which signal rises first. The READ_MODE parameter in the SMC_MODE register of the corresponding chip select indicates which signal of NRD and NCS controls the read operation.

21.9.2.1 Read is Controlled by NRD (READ_MODE = 1):

Figure 21-11 shows the waveforms of a read operation of a typical asynchronous RAM. The read data is available t_{PACC} after the falling edge of NRD, and turns to 'Z' after the rising edge of NRD. In this case, the READ_MODE must be set to 1 (read is controlled by NRD), to indicate that data is available with the rising edge of NRD. The SMC samples the read data internally on the rising edge of Master Clock that generates the rising edge of NRD, whatever the programmed waveform of NCS may be.

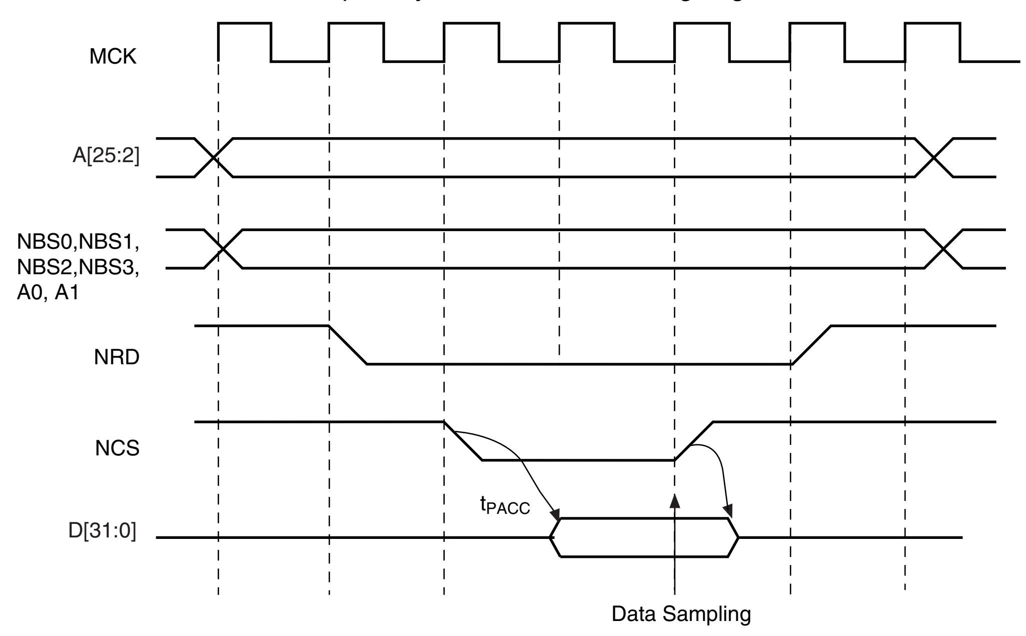
Figure 21-11. READ_MODE = 1: Data is sampled by SMC before the rising edge of NRD



21.9.2.2 Read is Controlled by NCS (READ_MODE = 0)

Figure 21-12 shows the typical read cycle of an LCD module. The read data is valid t_{PACC} after the falling edge of the NCS signal and remains valid until the rising edge of NCS. Data must be sampled when NCS is raised. In that case, the READ_MODE must be set to 0 (read is controlled by NCS): the SMC internally samples the data on the rising edge of Master Clock that generates the rising edge of NCS, whatever the programmed waveform of NRD may be.

Figure 21-12. READ_MODE = 0: Data is sampled by SMC before the rising edge of NCS



21.9.3 Write Waveforms

The write protocol is similar to the read protocol. It is depicted in Figure 21-13. The write cycle starts with the address setting on the memory address bus.

21.9.3.1 NWE Waveforms

The NWE signal is characterized by a setup timing, a pulse width and a hold timing.

a. NWE_SETUP: the NWE setup time is defined as the setup of address and data before the NWE falling edge;

b. NWE_PULSE: The NWE pulse length is the time between NWE falling edge and NWE rising edge;

c. NWE_HOLD: The NWE hold time is defined as the hold time of address and data after the NWE rising edge.

The NWE waveforms apply to all byte-write lines in Byte Write access mode: NWR0 to NWR3.

21.9.3.2 NCS Waveforms

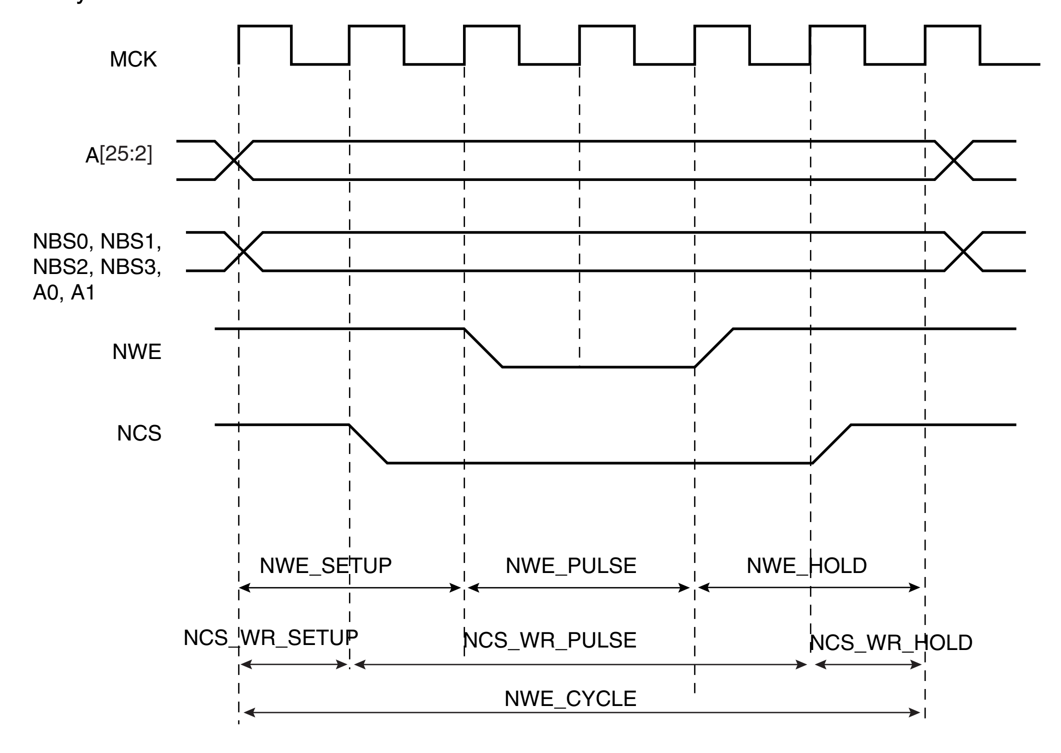
The NCS signal waveforms in write operation are not the same that those applied in read operations, but are separately defined:

a. NCS_WR_SETUP: the NCS setup time is defined as the setup time of address before the NCS falling edge.

b. NCS_WR_PULSE: the NCS pulse length is the time between NCS falling edge and NCS rising edge;

c. NCS_WR_HOLD: the NCS hold time is defined as the hold time of address after the NCS rising edge.

Figure 21-13. Write Cycle



21.9.3.3 Write Cycle

The write_cycle time is defined as the total duration of the write cycle, that is, from the time where address is set on the address bus to the point where address may change. The total write cycle time is equal to:

NWE_CYCLE = NWE_SETUP + NWE_PULSE + NWE_HOLD

= NCS_WR_SETUP + NCS_WR_PULSE + NCS_WR_HOLD

All NWE and NCS (write) timings are defined separately for each chip select as an integer number of Master Clock cycles. To ensure that the NWE and NCS timings are coherent, the user must define the total write cycle instead of the hold timing. This implicitly defines the NWE hold time and NCS (write) hold times as:

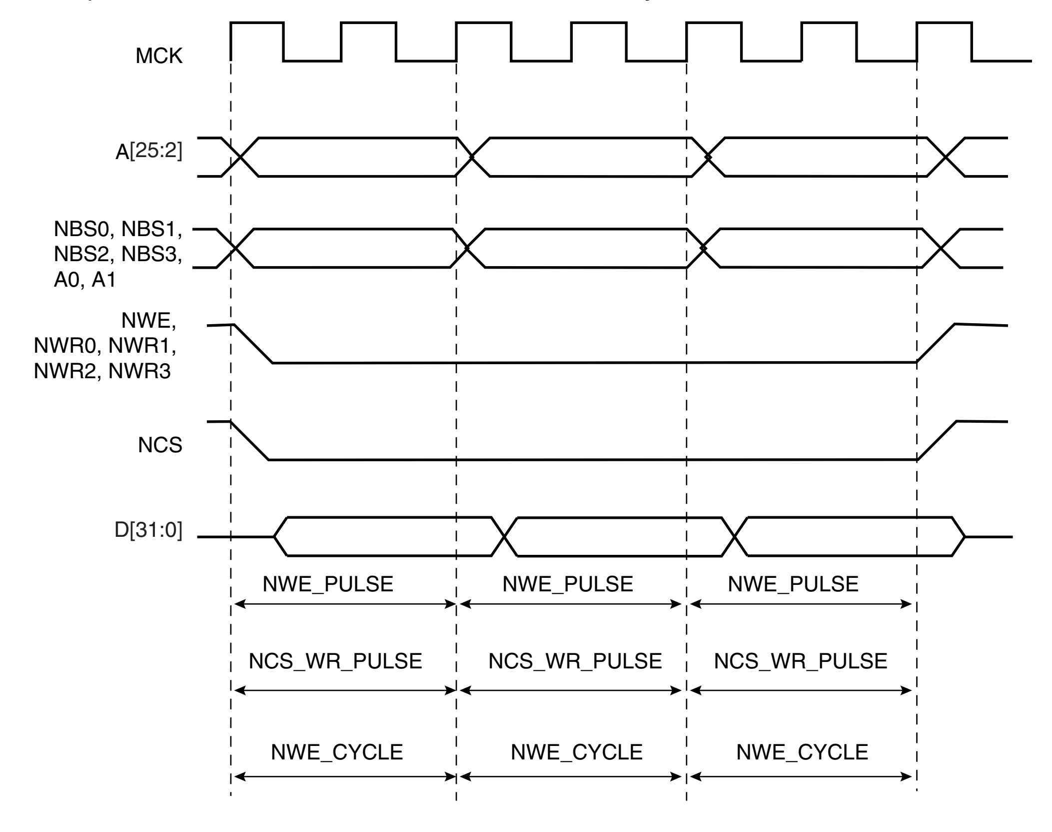
NWE_HOLD = NWE_CYCLE - NWE_SETUP - NWE_PULSE

NCS_WR_HOLD = NWE_CYCLE - NCS_WR_SETUP - NCS_WR_PULSE

21.9.3.4 Null Delay Setup and Hold

If null setup parameters are programmed for NWE and/or NCS, NWE and/or NCS remain active continuously in case of consecutive write cycles in the same memory (see Figure 21-14). However, for devices that perform write operations on the rising edge of NWE or NCS, such as SRAM, either a setup or a hold must be programmed.

Figure 21-14. Null Setup and Hold Values of NCS and NWE in Write Cycle



21.9.3.5 Null Pulse

Programming null pulse is not permitted. Pulse must be at least set to 1. A null value leads to unpredictable behavior.

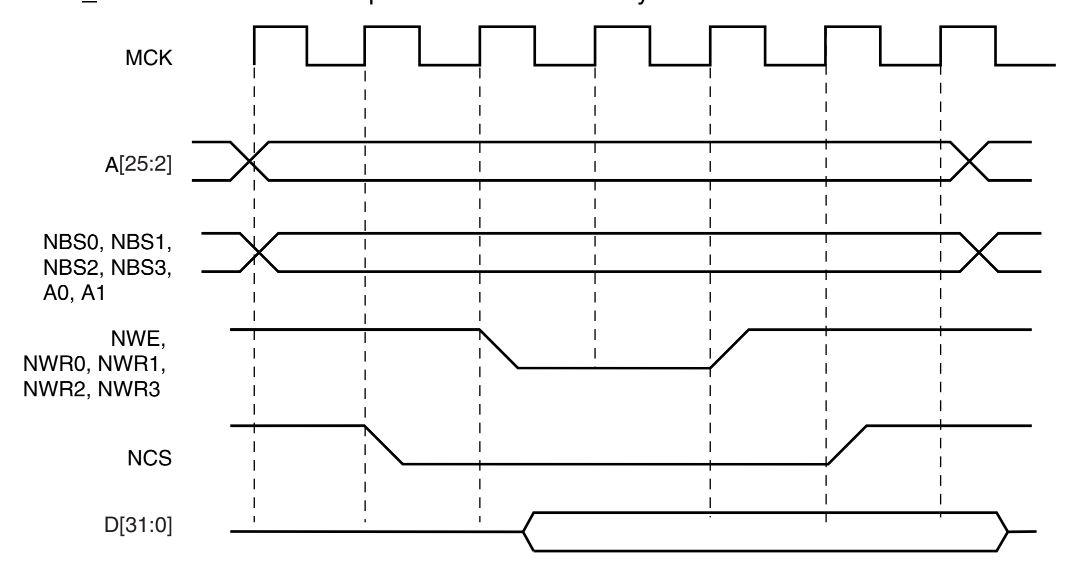
21.9.4 Write Mode

The WRITE_MODE parameter in the SMC_MODE register of the corresponding chip select indicates which signal controls the write operation.

21.9.4.1 Write is Controlled by NWE (WRITE_MODE = 1):

Figure 21-15 shows the waveforms of a write operation with WRITE_MODE set to 1. The data is put on the bus during the pulse and hold steps of the NWE signal. The internal data buffers are turned out after the NWE_SETUP time, and until the end of the write cycle, regardless of the programmed waveform on NCS.

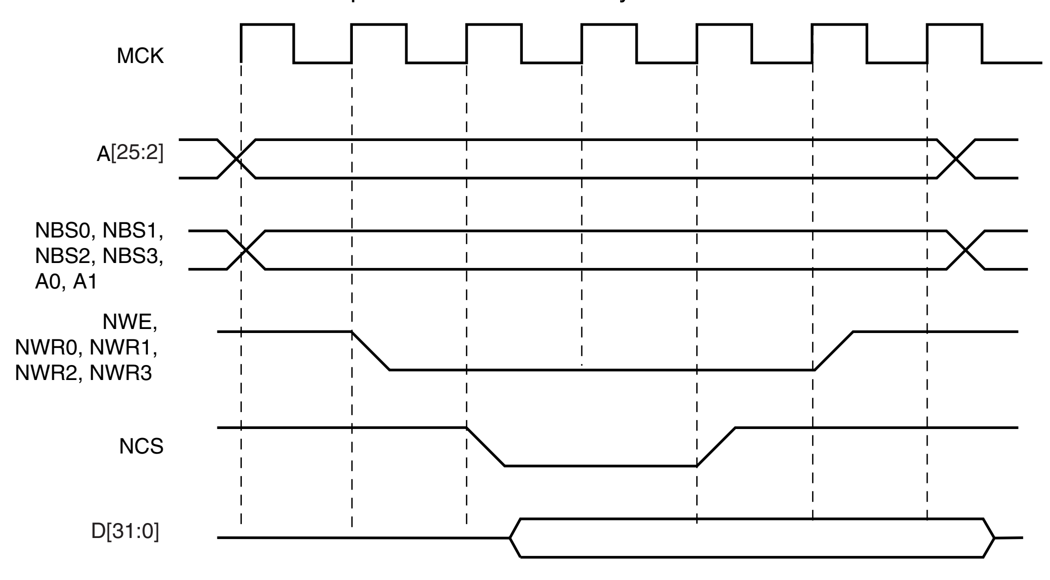
Figure 21-15. WRITE_MODE = 1. The write operation is controlled by NWE



21.9.4.2 Write is Controlled by NCS (WRITE_MODE = 0)

Figure 21-16 shows the waveforms of a write operation with WRITE_MODE set to 0. The data is put on the bus during the pulse and hold steps of the NCS signal. The internal data buffers are turned out after the NCS_WR_SETUP time, and until the end of the write cycle, regardless of the programmed waveform on NWE.

Figure 21-16. WRITE_MODE = 0. The write operation is controlled by NCS



21.9.5 Coding Timing Parameters

All timing parameters are defined for one chip select and are grouped together in one SMC_REGISTER according to their type.

The SMC_SETUP register groups the definition of all setup parameters:

- NRD_SETUP, NCS_RD_SETUP, NWE_SETUP, NCS_WR_SETUP

The SMC_PULSE register groups the definition of all pulse parameters:

- NRD_PULSE, NCS_RD_PULSE, NWE_PULSE, NCS_WR_PULSE

The SMC_CYCLE register groups the definition of all cycle parameters:

- NRD_CYCLE, NWE_CYCLE

Table 21-4 shows how the timing parameters are coded and their permitted range.

Table 21-4. Coding and Range of Timing Parameters

Coded Value	Number of Bits	Effective Value	Permitted Range	
			Coded Value	Effective Value
setup [5:0]	6	128 x setup[5] + setup[4:0]	0 ≤≤31	128 ≤≤128+31
pulse [6:0]	7	256 x pulse[6] + pulse[5:0]	0 ≤≤63	256 ≤≤256+63
cycle [8:0]	9	256 x cycle[8:7] + cycle[6:0]	0 ≤≤127	256 ≤≤256+127 512 ≤≤512+127 768 ≤≤768+127

21.9.6 Reset Values of Timing Parameters

Table 21-5 gives the default value of timing parameters at reset.

Table 21-5. Reset Values of Timing Parameters

Register	Reset Value	
SMC_SETUP	0x01010101	All setup timings are set to 1
SMC_PULSE	0x01010101	All pulse timings are set to 1
SMC_CYCLE	0x00030003	The read and write operation last 3 Master Clock cycles and provide one hold cycle
WRITE_MODE	1	write is controlled with NWE
READ_MODE	1	read is controlled with NRD

21.9.7 Usage Restriction

The SMC does not check the validity of the user-programmed parameters. If the sum of SETUP and PULSE parameters is larger than the corresponding CYCLE parameter, this leads to unpredictable behavior of the SMC.

For read operations:

Null but positive setup and hold of address and NRD and/or NCS can not be guaranteed at the memory interface because of the propagation delay of theses signals through external logic and pads. If positive setup and hold values must be verified, then it is strictly recommended to program non-null values so as to cover possible skews between address, NCS and NRD signals.

For write operations:

If a null hold value is programmed on NWE, the SMC can guarantee a positive hold of address, byte select lines, and NCS signal after the rising edge of NWE. This is true for WRITE_MODE = 1 only. See ”Early Read Wait State” on page 162.

For read and write operations: a null value for pulse parameters is forbidden and may lead to unpredictable behavior.

In read and write cycles, the setup and hold time parameters are defined in reference to the address bus. For external devices that require setup and hold time between NCS and NRD signals (read), or between NCS and NWE signals (write), these setup and hold times must be converted into setup and hold times in reference to the address bus.

21.10 Automatic Wait States

Under certain circumstances, the SMC automatically inserts idle cycles between accesses to avoid bus contention or operation conflict.

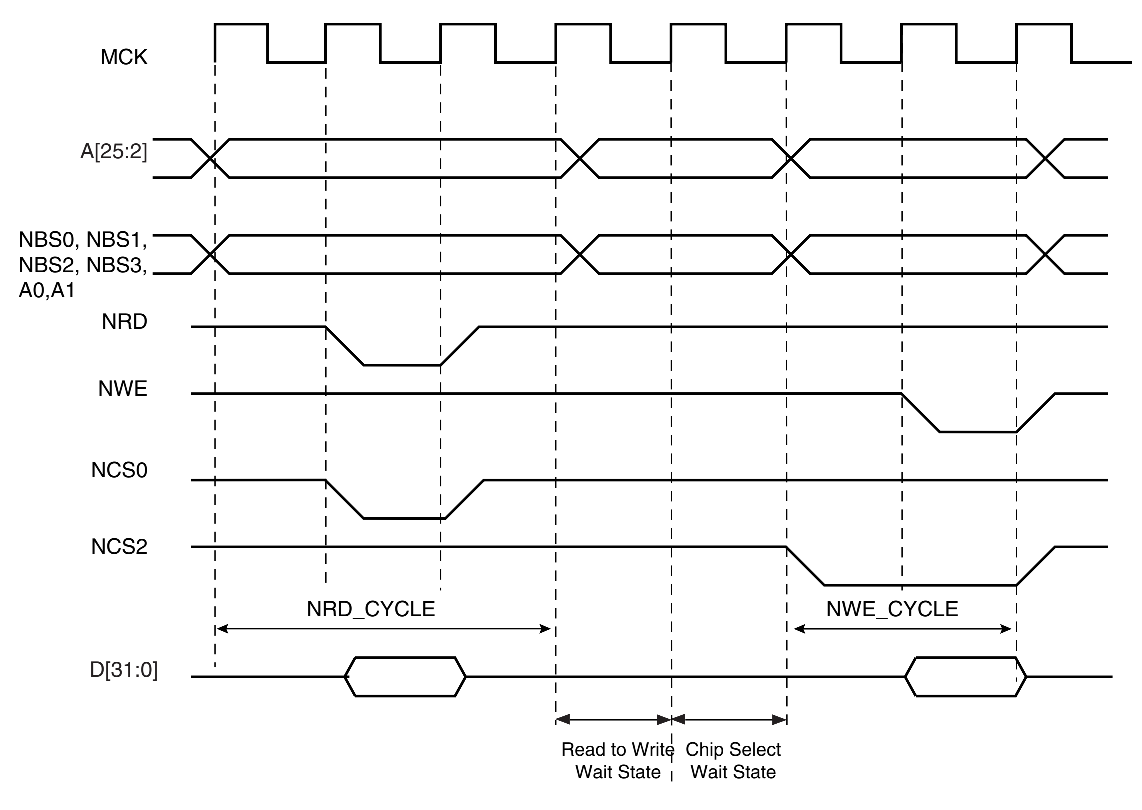
21.10.1 Chip Select Wait States

The SMC always inserts an idle cycle between 2 transfers on separate chip selects. This idle cycle ensures that there is no bus contention between the de-activation of one device and the activation of the next one.

During chip select wait state, all control lines are turned inactive: NBS0 to NBS3, NWR0 to NWR3, NCS[0..7], NRD lines are all set to 1.

Figure 21-17 illustrates a chip select wait state between access on Chip Select 0 and Chip Select 2.

Figure 21-17. Chip Select Wait State between a Read Access on NCS0 and a Write Access on NCS2



21.10.2 Early Read Wait State

In some cases, the SMC inserts a wait state cycle between a write access and a read access to allow time for the write cycle to end before the subsequent read cycle begins. This wait state is not generated in addition to a chip select wait state. The early read cycle thus only occurs between a write and read access to the same memory device (same chip select).

An early read wait state is automatically inserted if at least one of the following conditions is valid:

- if the write controlling signal has no hold time and the read controlling signal has no setup time (Figure 21-18).
- in NCS write controlled mode (WRITE_MODE = 0), if there is no hold timing on the NCS signal and the NCS_RD_SETUP parameter is set to 0, regardless of the read mode (Figure 21-19). The write operation must end with a NCS rising edge. Without an Early Read Wait State, the write operation could not complete properly.
- in NWE controlled mode (WRITE_MODE = 1) and if there is no hold timing (NWE_HOLD = 0), the feedback of the write control signal is used to control address, data, chip select and byte select lines. If the external write control signal is not inactivated as expected due to load capacitances, an Early Read Wait State is inserted and address, data and control signals are maintained one more cycle. See Figure 21-20.

Figure 21-18. Early Read Wait State: Write with No Hold Followed by Read with No Setup

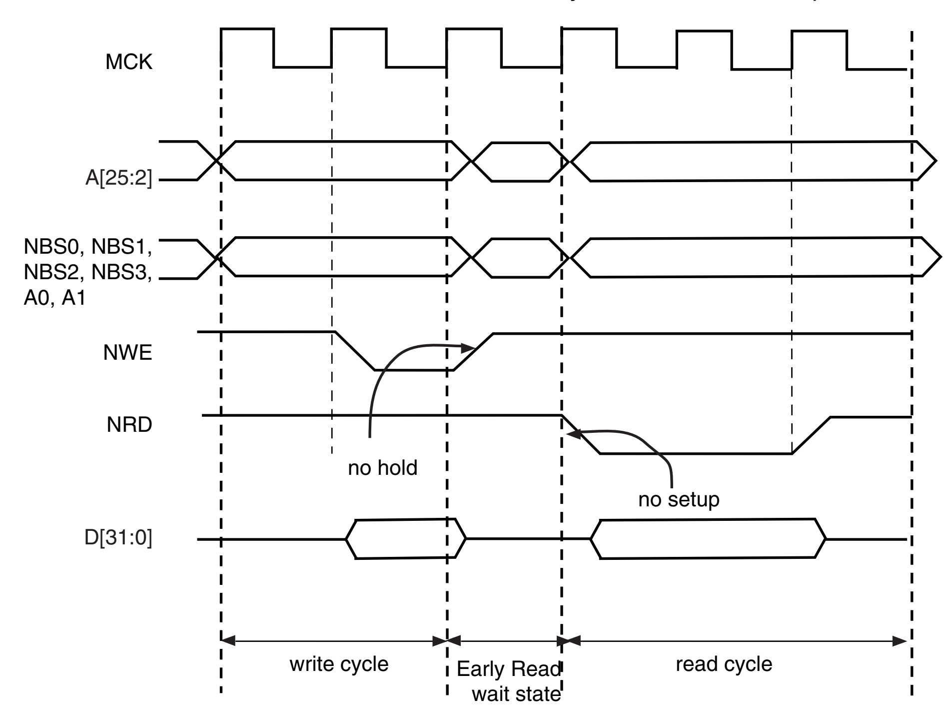


Figure 21-19. Early Read Wait State: NCS Controlled Write with No Hold Followed by a Read with No NCS Setup

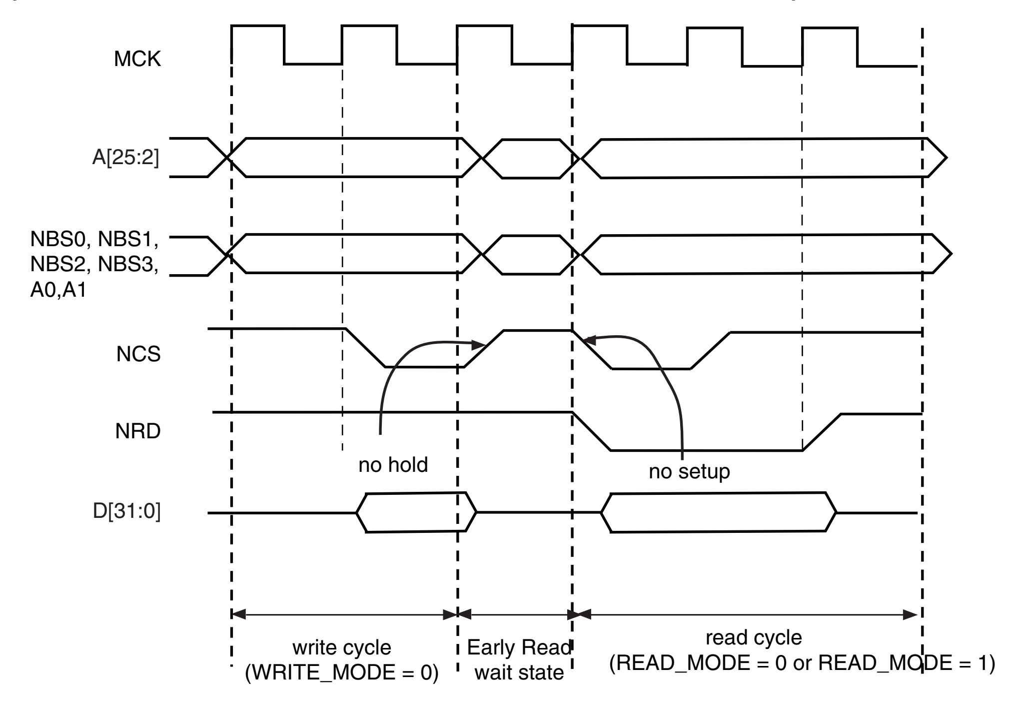
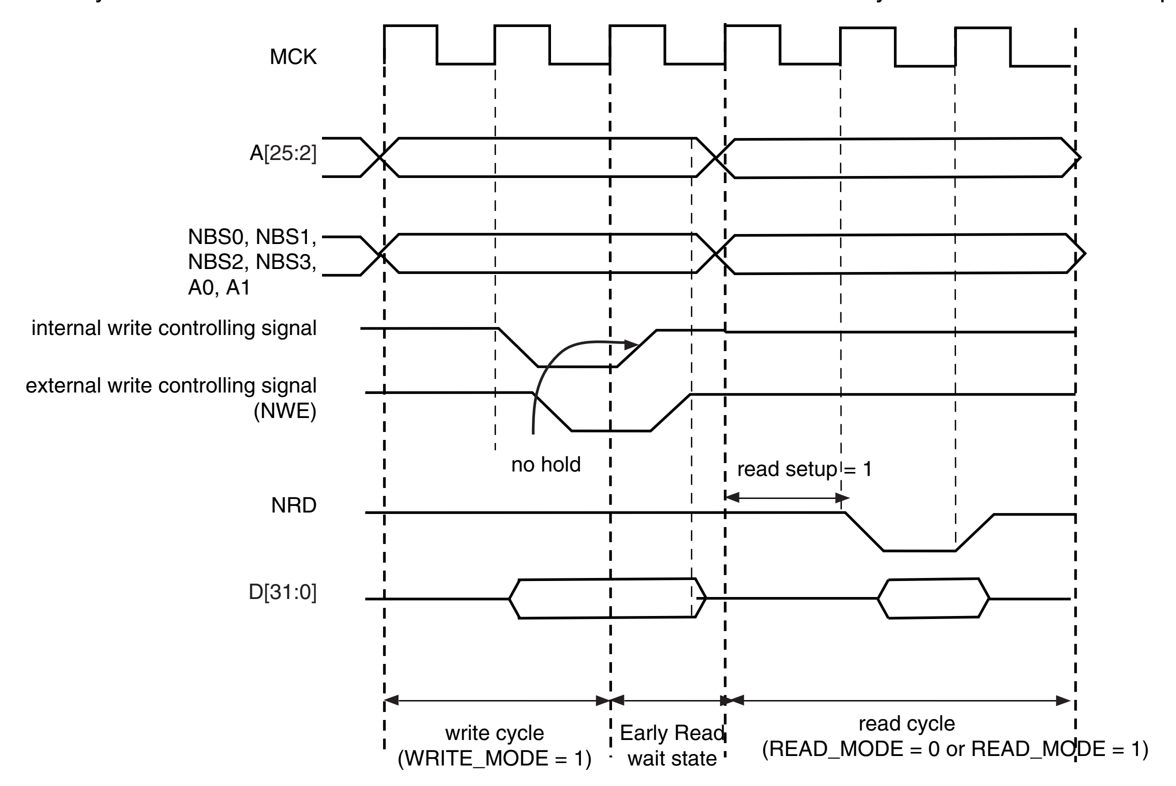


Figure 21-20. Early Read Wait State: NWE-controlled Write with No Hold Followed by a Read with one Set-up Cycle



21.10.3 Reload User Configuration Wait State

The user may change any of the configuration parameters by writing the SMC user interface.

When detecting that a new user configuration has been written in the user interface, the SMC inserts a wait state before starting the next access. The so called "Reload User Configuration Wait State" is used by the SMC to load the new set of parameters to apply to next accesses.

The Reload Configuration Wait State is not applied in addition to the Chip Select Wait State. If accesses before and after re-programming the user interface are made to different devices (Chip Selects), then one single Chip Select Wait State is applied.

On the other hand, if accesses before and after writing the user interface are made to the same device, a Reload Configuration Wait State is inserted, even if the change does not concern the current Chip Select.

21.10.3.1 User Procedure

To insert a Reload Configuration Wait State, the SMC detects a write access to any SMC_MODE register of the user interface. If the user only modifies timing registers (SMC_SETUP, SMC_PULSE, SMC_CYCLE registers) in the user interface, he must validate the modification by writing the SMC_MODE, even if no change was made on the mode parameters.

21.10.3.2 Slow Clock Mode Transition

A Reload Configuration Wait State is also inserted when the Slow Clock Mode is entered or exited, after the end of the current transfer (see "Slow Clock Mode" on page 175).

21.10.4 Read to Write Wait State

Due to an internal mechanism, a wait cycle is always inserted between consecutive read and write SMC accesses.

This wait cycle is referred to as a read to write wait state in this document.

This wait cycle is applied in addition to chip select and reload user configuration wait states when they are to be inserted. See Figure 21-17 on page 162.

21.11 Data Float Wait States

Some memory devices are slow to release the external bus. For such devices, it is necessary to add wait states (data float wait states) after a read access:

- before starting a read access to a different external memory
- before starting a write access to the same device or to a different external one.

The Data Float Output Time (t_{DF}) for each external memory device is programmed in the TDF_CYCLES field of the SMC_MODE register for the corresponding chip select. The value of TDF_CYCLES indicates the number of data float wait cycles (between 0 and 15) before the external device releases the bus, and represents the time allowed for the data output to go to high impedance after the memory is disabled.

Data float wait states do not delay internal memory accesses. Hence, a single access to an external memory with long t_{DF} will not slow down the execution of a program from internal memory.

The data float wait states management depends on the READ_MODE and the TDF_MODE fields of the SMC_MODE register for the corresponding chip select.

21.11.1 READ_MODE

Setting the READ_MODE to 1 indicates to the SMC that the NRD signal is responsible for turning off the tri-state buffers of the external memory device. The Data Float Period then begins after the rising edge of the NRD signal and lasts TDF_CYCLES MCK cycles.

When the read operation is controlled by the NCS signal (READ_MODE = 0), the TDF field gives the number of MCK cycles during which the data bus remains busy after the rising edge of NCS.

Figure 21-21 illustrates the Data Float Period in NRD-controlled mode (READ_MODE =1), assuming a data float period of 2 cycles (TDF_CYCLES = 2). Figure 21-22 shows the read operation when controlled by NCS (READ_MODE = 0) and the TDF_CYCLES parameter equals 3.



Figure 21-21. TDF Period in NRD Controlled Read Access (TDF = 2)

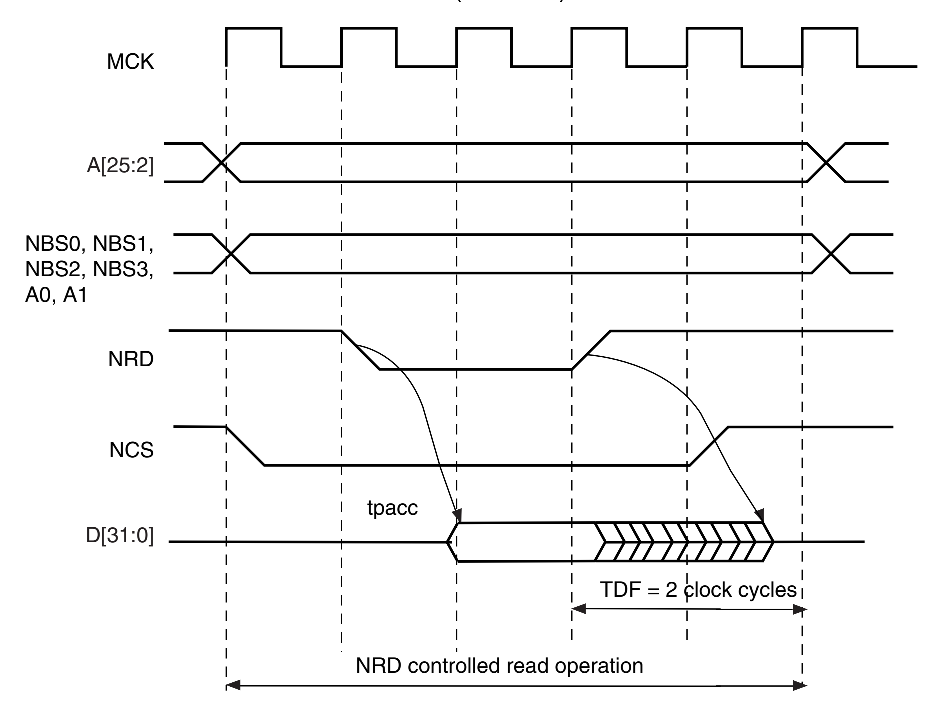
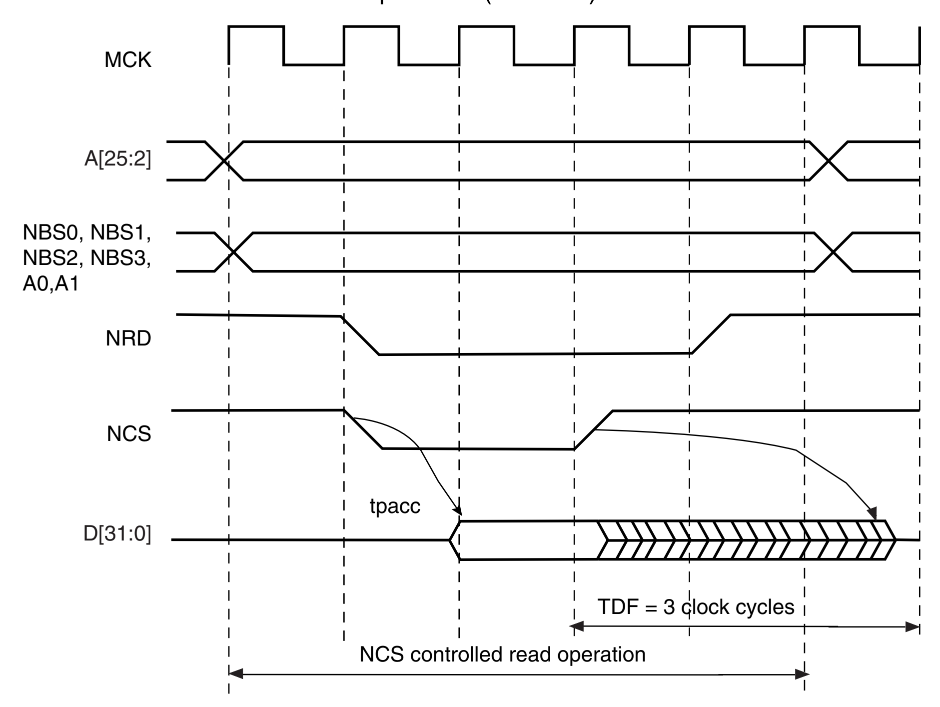


Figure 21-22. TDF Period in NCS Controlled Read Operation (TDF = 3)



21.11.2 TDF Optimization Enabled (TDF_MODE = 1)

When the TDF_MODE of the SMC_MODE register is set to 1 (TDF optimization is enabled), the SMC takes advantage of the setup period of the next access to optimize the number of wait states cycle to insert.

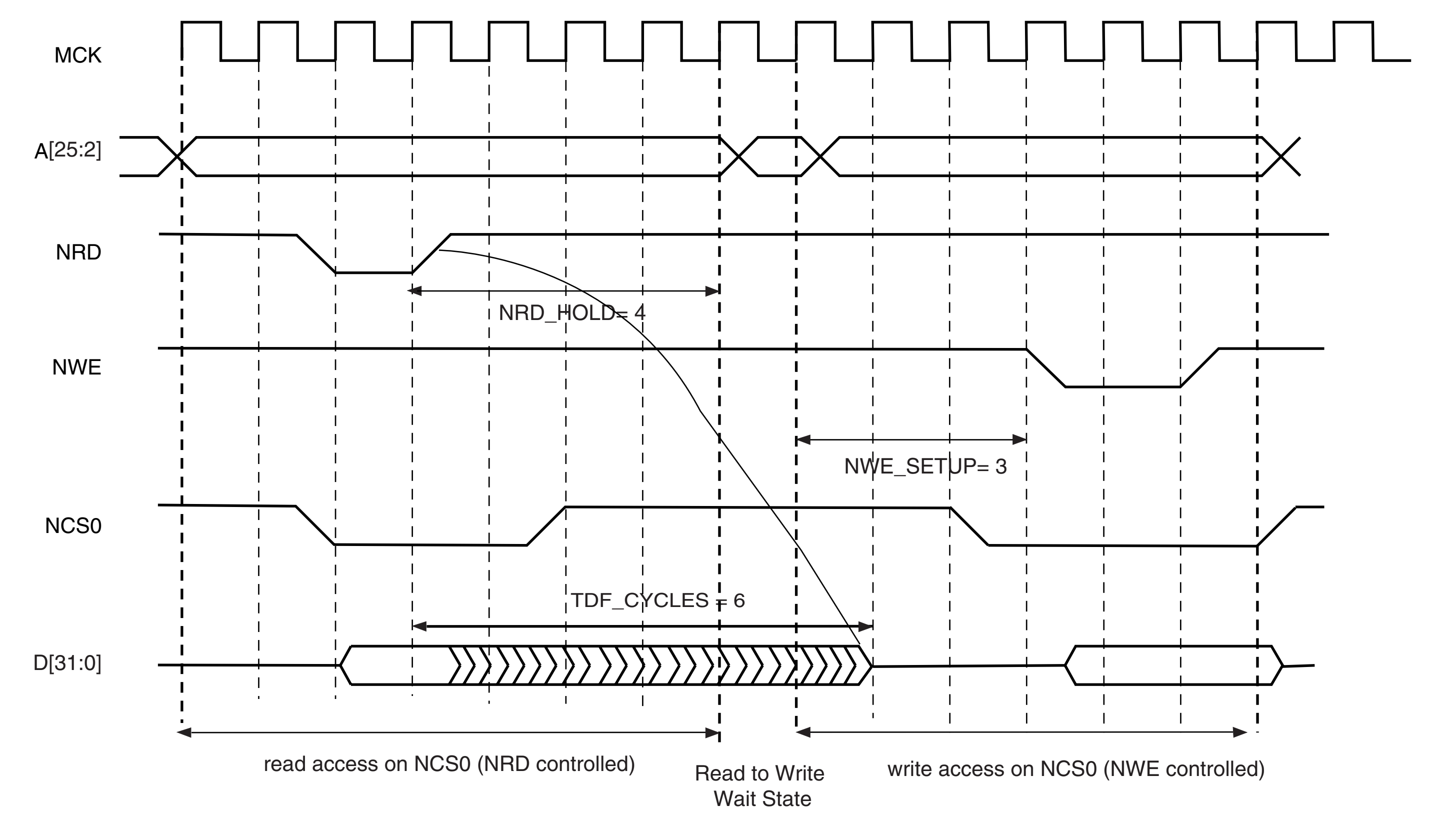
Figure 21-23 shows a read access controlled by NRD, followed by a write access controlled by NWE, on Chip Select 0. Chip Select 0 has been programmed with:

NRD_HOLD = 4; READ_MODE = 1 (NRD controlled)

NWE_SETUP = 3; WRITE_MODE = 1 (NWE controlled)

TDF_CYCLES = 6; TDF_MODE = 1 (optimization enabled).

Figure 21-23. TDF Optimization: No TDF wait states are inserted if the TDF period is over when the next access begins



21.11.3 TDF Optimization Disabled (TDF_MODE = 0)

When optimization is disabled, tdf wait states are inserted at the end of the read transfer, so that the data float period is ended when the second access begins. If the hold period of the read1 controlling signal overlaps the data float period, no additional tdf wait states will be inserted.

Figure 21-24, Figure 21-25 and Figure 21-26 illustrate the cases:

- read access followed by a read access on another chip select,
- read access followed by a write access on another chip select,
- read access followed by a write access on the same chip select,

with no TDF optimization.

Figure 21-24. TDF Optimization Disabled (TDF Mode = 0). TDF wait states between 2 read accesses on different chip selects

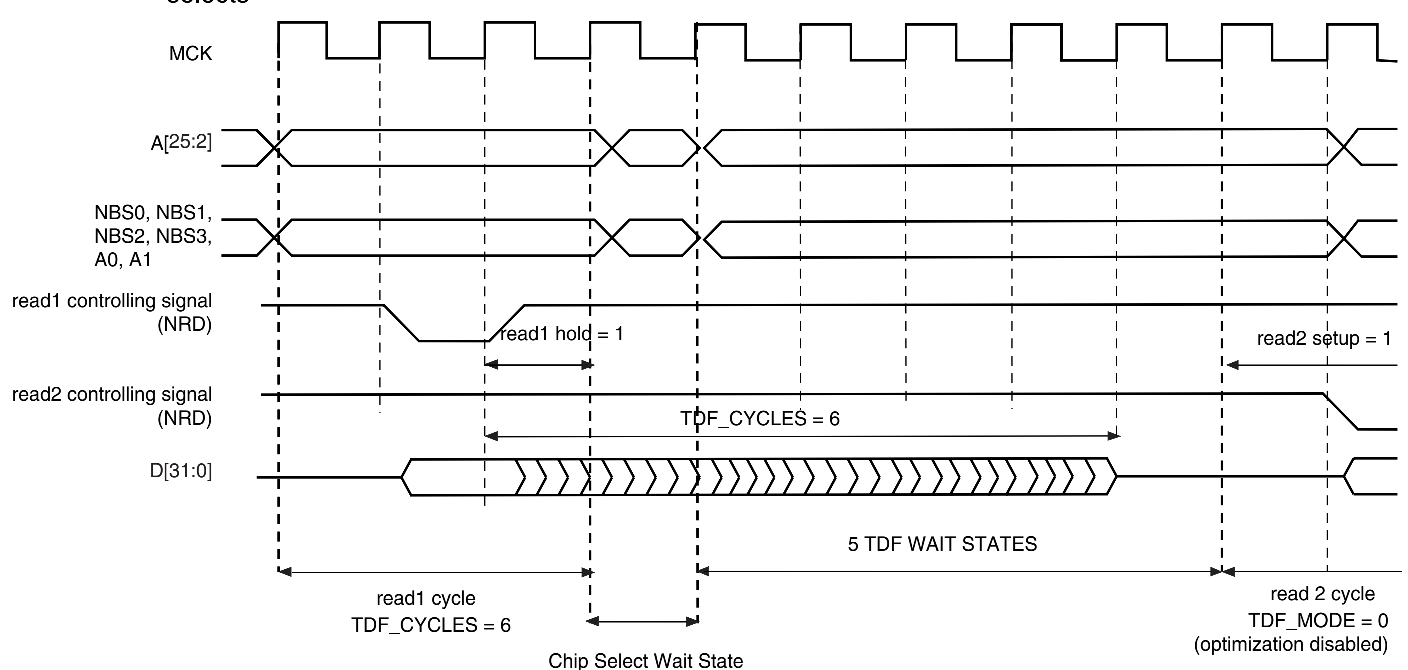


Figure 21-25. TDF Mode = 0: TDF wait states between a read and a write access on different chip selects

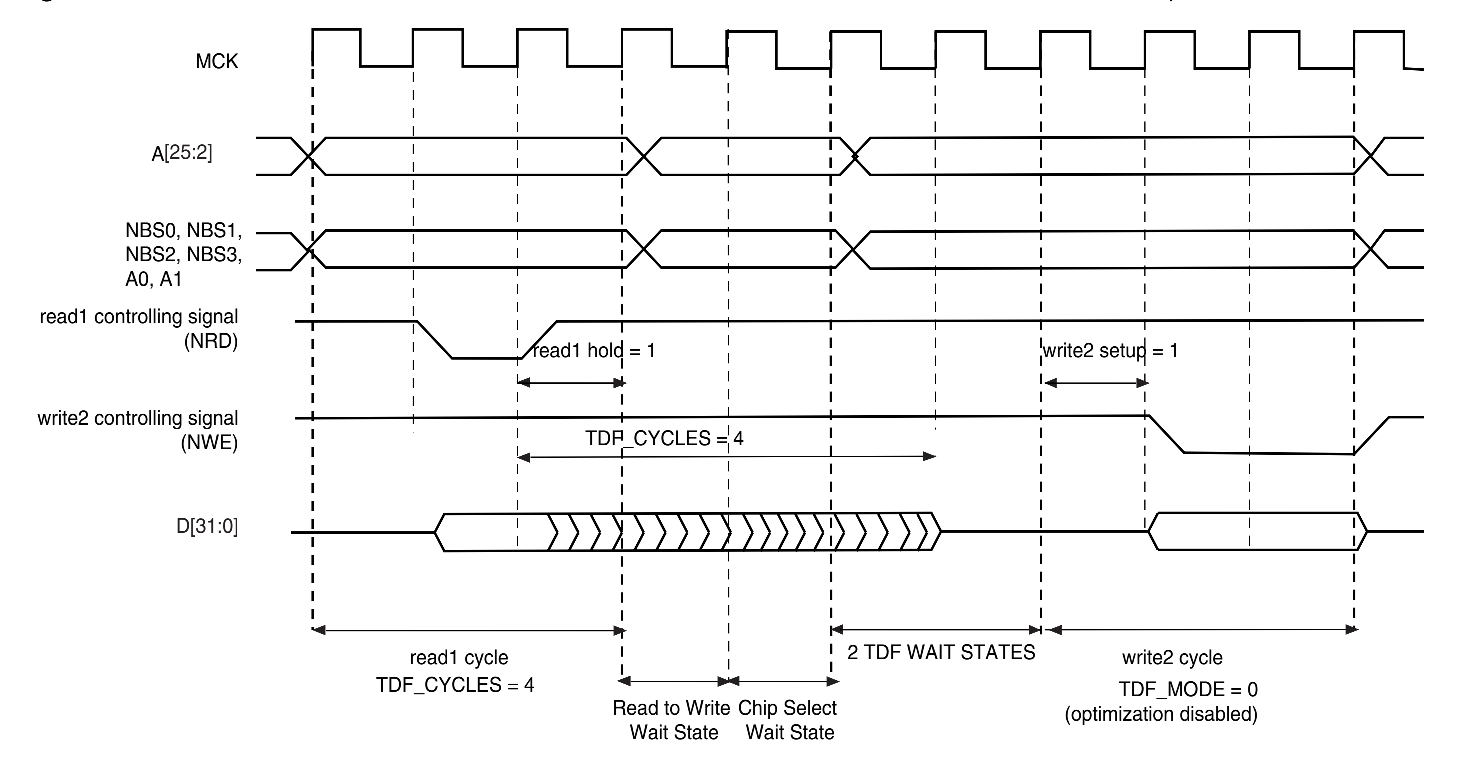
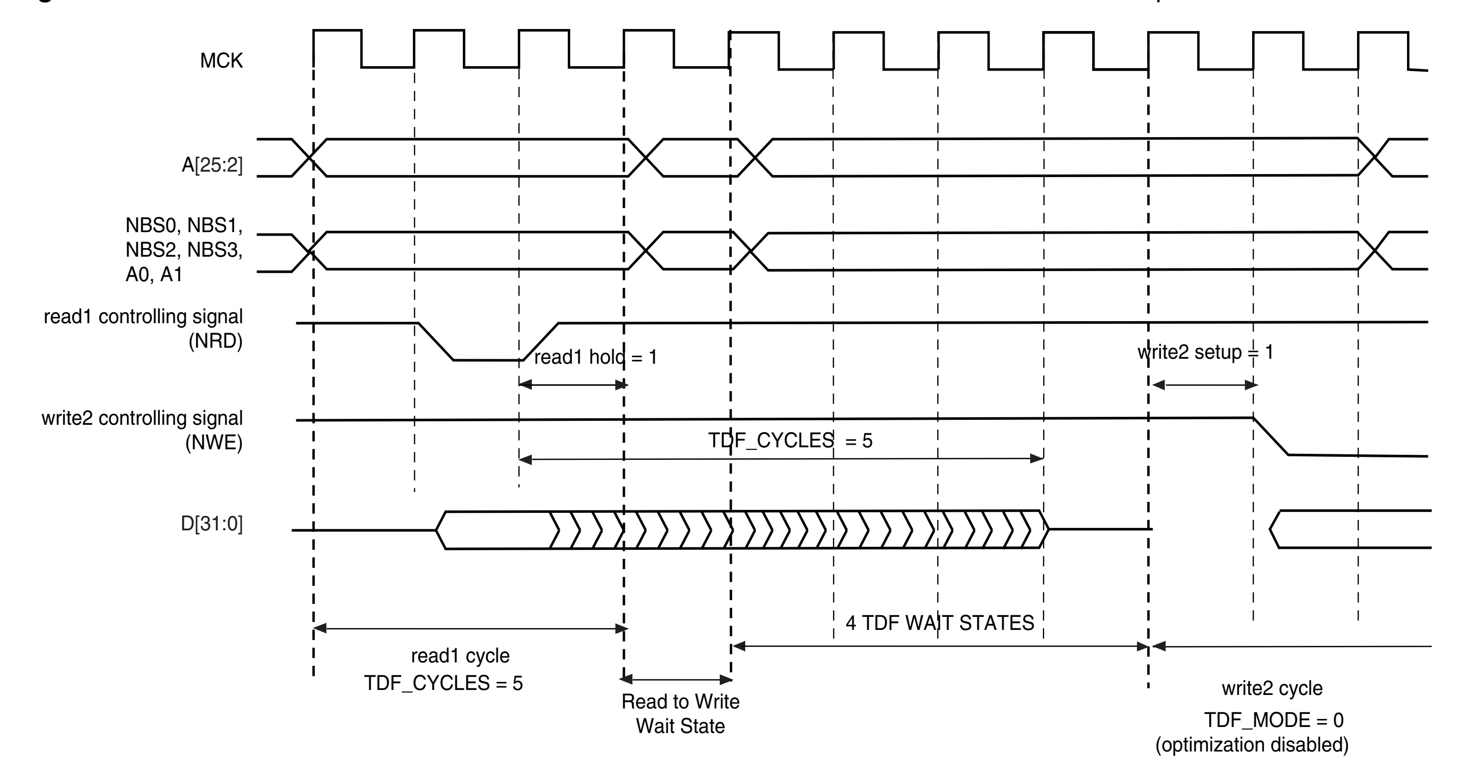


Figure 21-26. TDF Mode = 0: TDF wait states between read and write accesses on the same chip select



21.12 External Wait

Any access can be extended by an external device using the NWAIT input signal of the SMC. The EXNW_MODE field of the SMC_MODE register on the corresponding chip select must be set to either “10” (frozen mode) or “11” (ready mode). When the EXNW_MODE is set to “00” (disabled), the NWAIT signal is simply ignored on the corresponding chip select. The NWAIT signal delays the read or write operation in regards to the read or write controlling signal, depending on the read and write modes of the corresponding chip select.

21.12.1 Restriction

When one of the EXNW_MODE is enabled, **it is mandatory to program at least one hold cycle for the read/write controlling signal. For that reason, the NWAIT signal cannot be used in Page Mode (”Asynchronous Page Mode” on page 178), or in Slow Clock Mode (”Slow Clock Mode” on page 175).**

The NWAIT signal is assumed to be a response of the external device to the read/write request of the SMC. Then NWAIT is examined by the SMC only in the pulse state of the read or write controlling signal. The assertion of the NWAIT signal outside the expected period has no impact on SMC behavior.



21.12.2 Frozen Mode

When the external device asserts the NWAIT signal (active low), and after internal synchronization of this signal, the SMC state is frozen, i.e., SMC internal counters are frozen, and all control signals remain unchanged. When the resynchronized NWAIT signal is deasserted, the SMC completes the access, resuming the access from the point where it was stopped. See Figure 21-27. This mode must be selected when the external device uses the NWAIT signal to delay the access and to freeze the SMC.

The assertion of the NWAIT signal outside the expected period is ignored as illustrated in Figure 21-28.

Figure 21-27. Write Access with NWAIT Assertion in Frozen Mode (EXNW_MODE = 10)

EXNW_MODE = 10 (Frozen)
WRITE_MODE = 1 (NWE_controlled)

NWE_PULSE = 5
NCS_WR_PULSE = 7

Figure 21-28. Read Access with NWAIT Assertion in Frozen Mode (EXNW_MODE = 10)

MCK

A[25:2]

NBS0, NBS1, NBS2, NBS3, A0,A1

FROZEN STATE

NCS 4 3 2 2 2 1 0 2 1 0

NRD 1 0 5 5 5 4 3 2 1 0

NWAIT

internally synchronized NWAIT signal

Read cycle

EXNW_MODE = 10 (Frozen)
READ_MODE = 0 (NCS_controlled)

NRD_PULSE = 2, NRD_HOLD = 6
NCS_RD_PULSE =5, NCS_RD_HOLD =3

Assertion is ignored

21.12.3 Ready Mode

In Ready mode (EXNW_MODE = 11), the SMC behaves differently. Normally, the SMC begins the access by down counting the setup and pulse counters of the read/write controlling signal. In the last cycle of the pulse phase, the resynchronized NWAIT signal is examined.

If asserted, the SMC suspends the access as shown in Figure 21-29 and Figure 21-30. After deassertion, the access is completed: the hold step of the access is performed.

This mode must be selected when the external device uses deassertion of the NWAIT signal to indicate its ability to complete the read or write operation.

If the NWAIT signal is deasserted before the end of the pulse, or asserted after the end of the pulse of the controlling read/write signal, it has no impact on the access length as shown in Figure 21-30.

Figure 21-29. NWAIT Assertion in Write Access: Ready Mode (EXNW_MODE = 11)

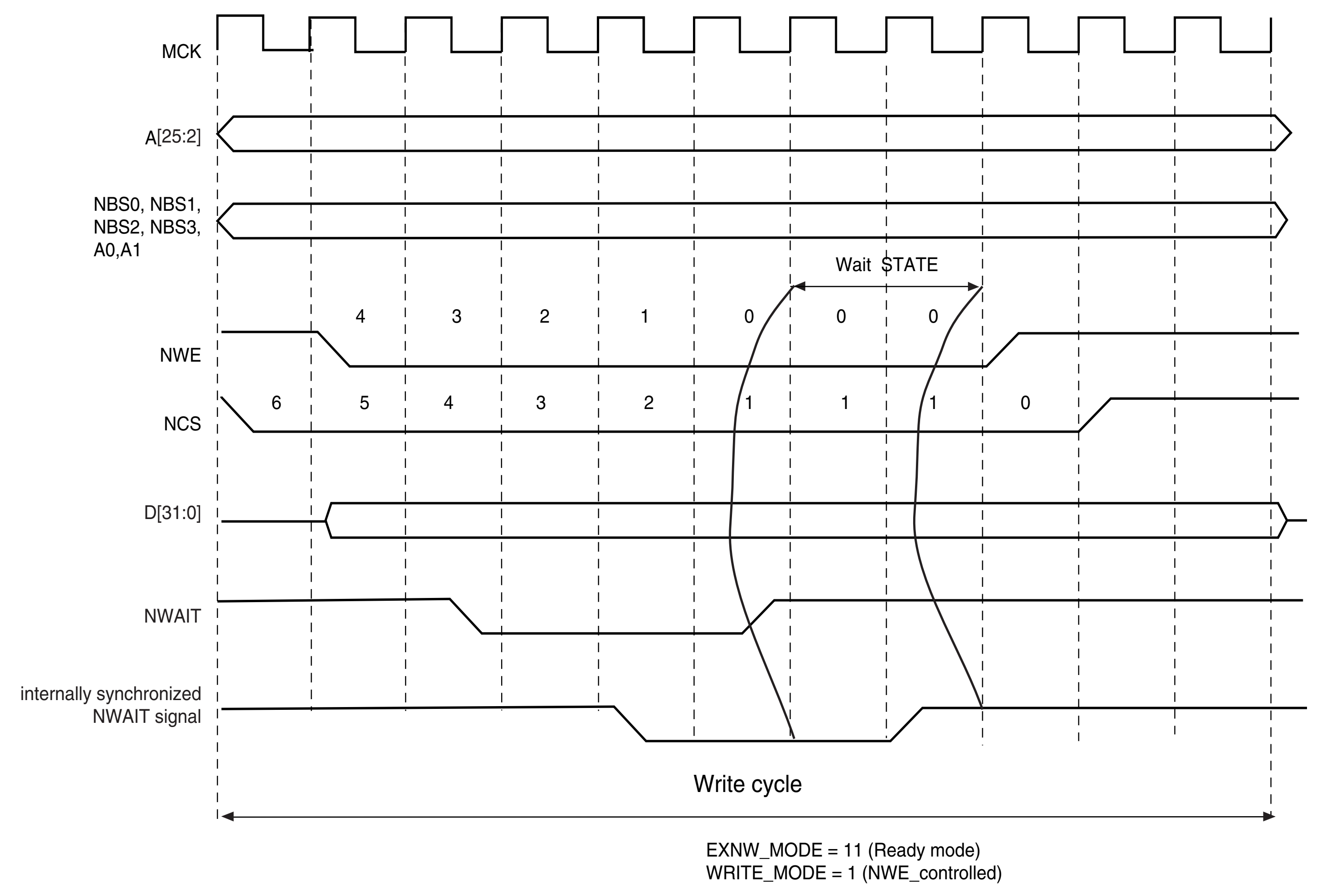


Figure 21-30. NWAIT Assertion in Read Access: Ready Mode (EXNW_MODE = 11)

MCK

A[25:2]

NBS0, NBS1, NBS2, NBS3, A0,A1

Wait STATE

NCS: 6 5 4 3 2 1 0 0

NRD: 6 5 4 3 2 1 1 0

NWAIT

internally synchronized NWAIT signal

Read cycle

EXNW_MODE = 11(Ready mode)
READ_MODE = 0 (NCS_controlled)

NRD_PULSE = 7
NCS_RD_PULSE =7

Assertion is ignored

Assertion is ignored

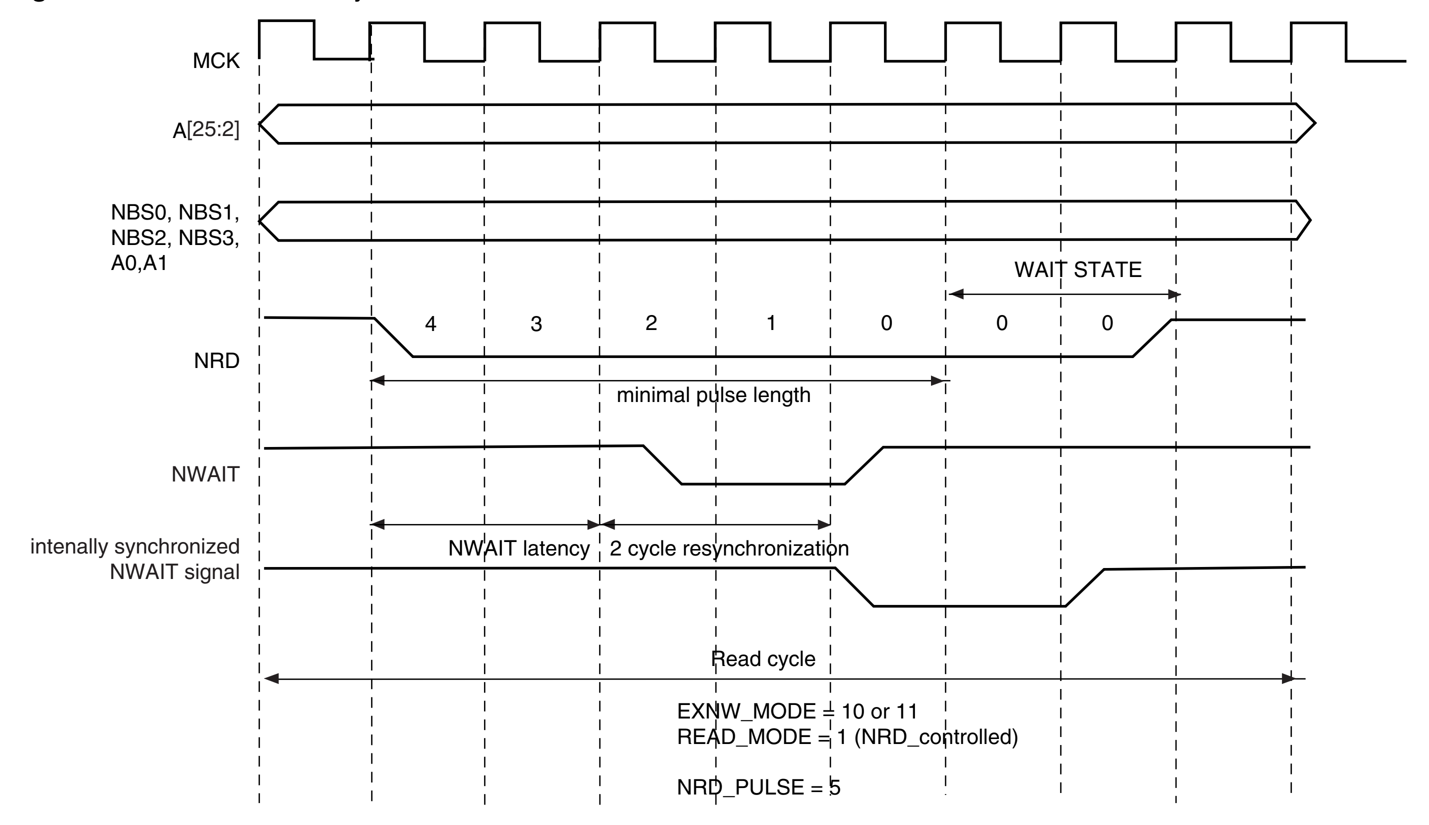
21.12.4 NWAIT Latency and Read/write Timings

There may be a latency between the assertion of the read/write controlling signal and the assertion of the NWAIT signal by the device. The programmed pulse length of the read/write controlling signal must be at least equal to this latency plus the 2 cycles of resynchronization + 1 cycle. Otherwise, the SMC may enter the hold state of the access without detecting the NWAIT signal assertion. This is true in frozen mode as well as in ready mode. This is illustrated on Figure 21-31.

When EXNW_MODE is enabled (ready or frozen), the user must program a pulse length of the read and write controlling signal of at least:

minimal pulse length = NWAIT latency + 2 resynchronization cycles + 1 cycle

Figure 21-31. NWAIT Latency



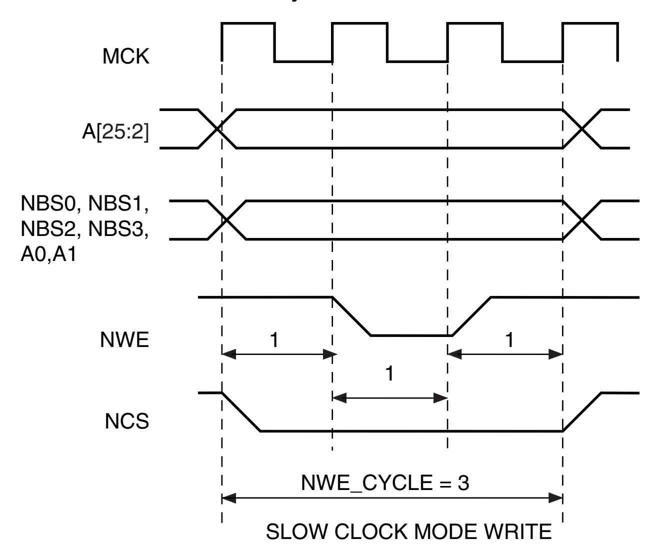
21.13 Slow Clock Mode

The SMC is able to automatically apply a set of “slow clock mode” read/write waveforms when an internal signal driven by the Power Management Controller is asserted because MCK has been turned to a very slow clock rate (typically 32kHz clock rate). In this mode, the user-programmed waveforms are ignored and the slow clock mode waveforms are applied. This mode is provided so as to avoid reprogramming the User Interface with appropriate waveforms at very slow clock rate. When activated, the slow mode is active on all chip selects.

21.13.1 Slow Clock Mode Waveforms

Figure 21-32 illustrates the read and write operations in slow clock mode. They are valid on all chip selects. Table 21-6 indicates the value of read and write parameters in slow clock mode.

Figure 21-32. Read/write Cycles in Slow Clock Mode



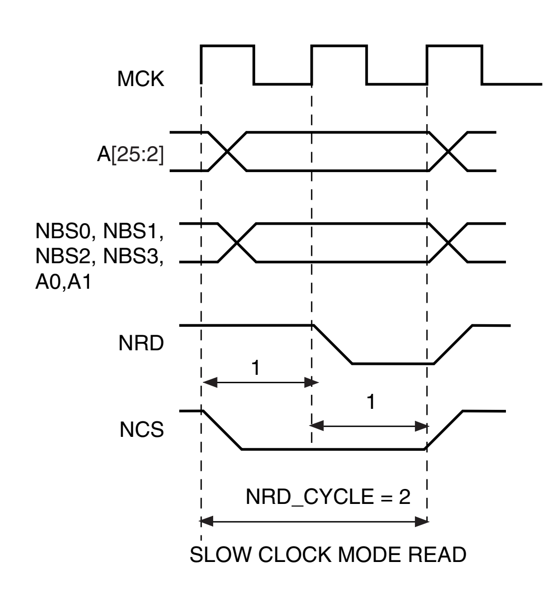


Table 21-6. Read and Write Timing Parameters in Slow Clock Mode

Read Parameters	Duration (cycles)	Write Parameters	Duration (cycles)
NRD_SETUP	1	NWE_SETUP	1
NRD_PULSE	1	NWE_PULSE	1
NCS_RD_SETUP	0	NCS_WR_SETUP	0
NCS_RD_PULSE	2	NCS_WR_PULSE	3
NRD_CYCLE	2	NWE_CYCLE	3

21.13.2 Switching from (to) Slow Clock Mode to (from) Normal Mode

When switching from slow clock mode to the normal mode, the current slow clock mode transfer is completed at high clock rate, with the set of slow clock mode parameters.See Figure 21-33 on page 176. The external device may not be fast enough to support such timings.

Figure 21-34 illustrates the recommended procedure to properly switch from one mode to the other.

Figure 21-33. Clock Rate Transition Occurs while the SMC is Performing a Write Operation

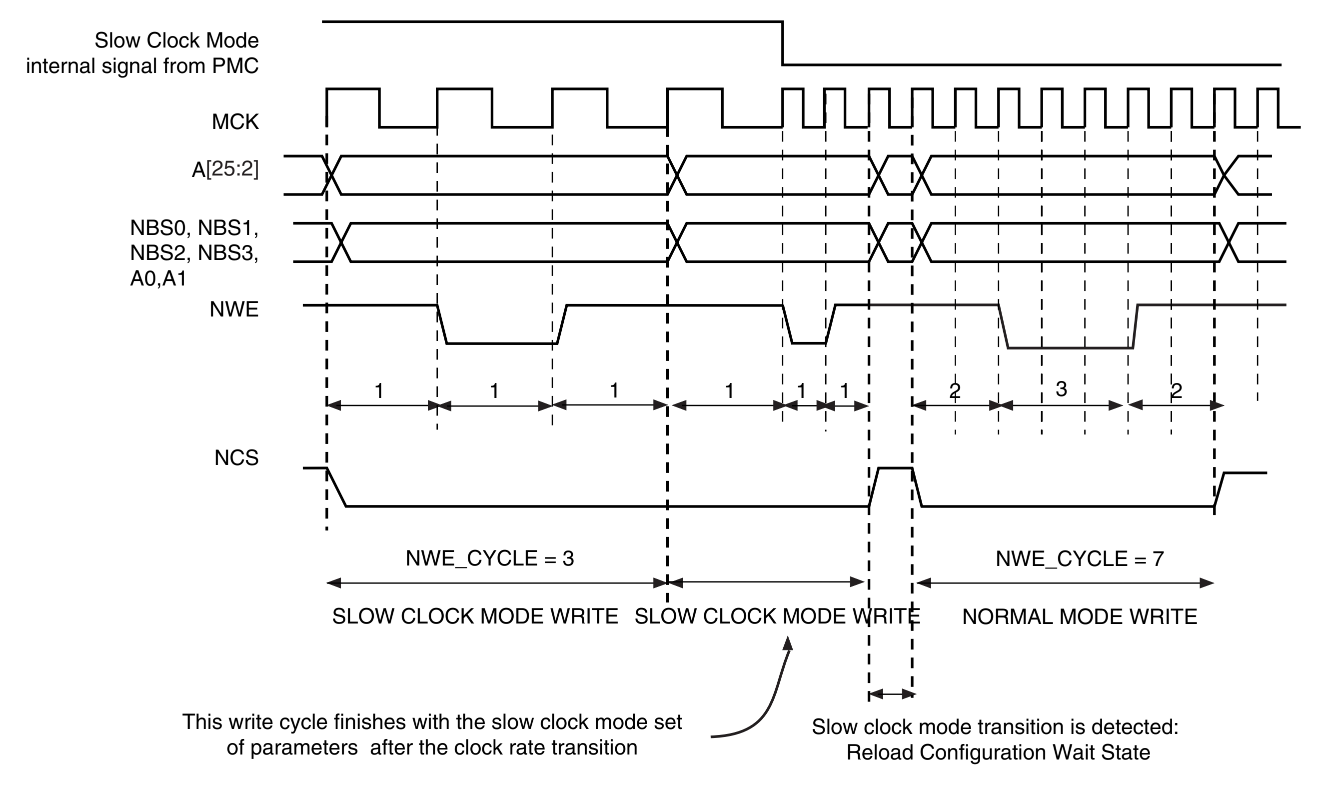
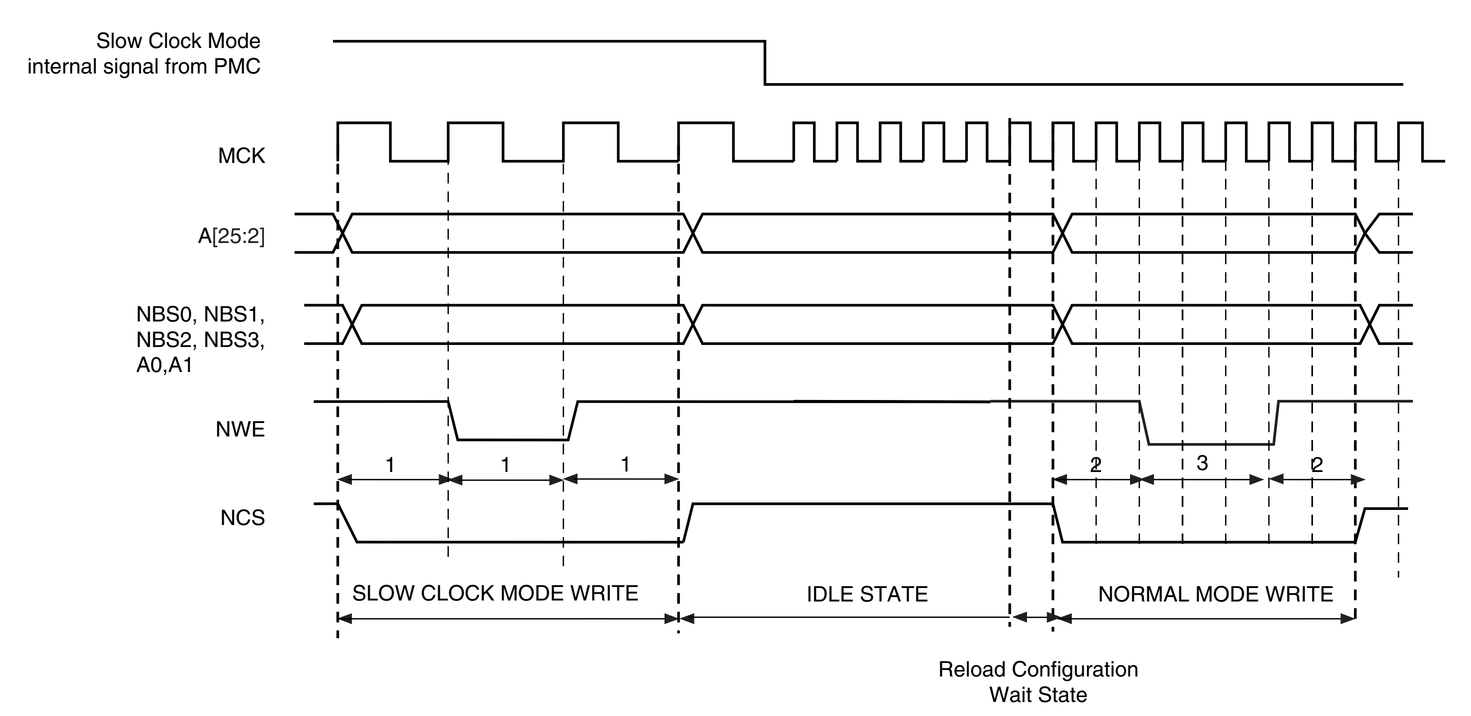


Figure 21-34. Recommended Procedure to Switch from Slow Clock Mode to Normal Mode or from Normal Mode to Slow Clock Mode



21.14 Asynchronous Page Mode

The SMC supports asynchronous burst reads in page mode, providing that the page mode is enabled in the SMC_MODE register (PMEN field). The page size must be configured in the SMC_MODE register (PS field) to 4, 8, 16 or 32 bytes.

The page defines a set of consecutive bytes into memory. A 4-byte page (resp. 8-, 16-, 32-byte page) is always aligned to 4-byte boundaries (resp. 8-, 16-, 32-byte boundaries) of memory. The MSB of data address defines the address of the page in memory, the LSB of address define the address of the data in the page as detailed in Table 21-7.

With page mode memory devices, the first access to one page (t_{pa}) takes longer than the subsequent accesses to the page (t_{sa}) as shown in Figure 21-35. When in page mode, the SMC enables the user to define different read timings for the first access within one page, and next accesses within the page.

Table 21-7. Page Address and Data Address within a Page

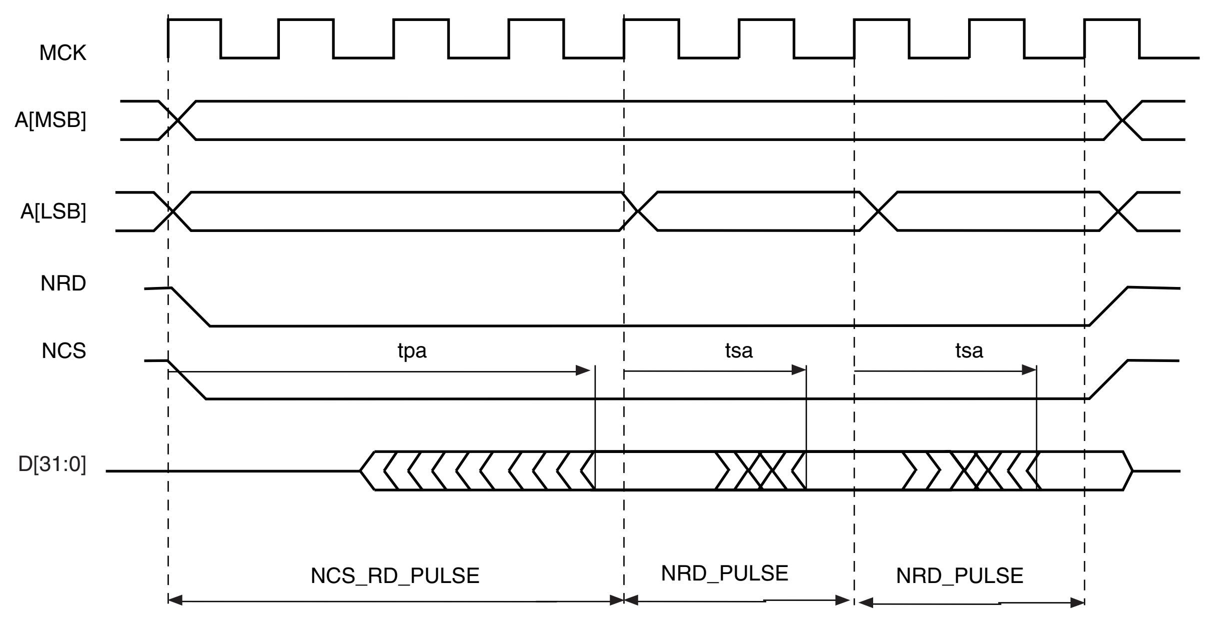
Page Size	Page Address(1)	Data Address in the Page(2)
4 bytes	A[25:2]	A[1:0]
8 bytes	A[25:3]	A[2:0]
16 bytes	A[25:4]	A[3:0]
32 bytes	A[25:5]	A[4:0]

Notes: 1. A denotes the address bus of the memory device
2. For 16-bit devices, the bit 0 of address is ignored. For 32-bit devices, bits [1:0] are ignored.

21.14.1 Protocol and Timings in Page Mode

Figure 21-35 shows the NRD and NCS timings in page mode access.

Figure 21-35. Page Mode Read Protocol (Address MSB and LSB are defined in Table 21-7)



The NRD and NCS signals are held low during all read transfers, whatever the programmed values of the setup and hold timings in the User Interface may be. Moreover, the NRD and NCS timings are identical. The pulse length of the first access to the page is defined with the

NCS_RD_PULSE field of the SMC_PULSE register. The pulse length of subsequent accesses within the page are defined using the NRD_PULSE parameter.

In page mode, the programming of the read timings is described in Table 21-8:

Table 21-8. Programming of Read Timings in Page Mode

Parameter	Value	Definition
READ_MODE	'x'	No impact
NCS_RD_SETUP	'x'	No impact
NCS_RD_PULSE	t_{pa}	Access time of first access to the page
NRD_SETUP	'x'	No impact
NRD_PULSE	t_{sa}	Access time of subsequent accesses in the page
NRD_CYCLE	'x'	No impact

The SMC does not check the coherency of timings. It will always apply the NCS_RD_PULSE timings as page access timing (t_{pa}) and the NRD_PULSE for accesses to the page (t_{sa}), even if the programmed value for t_{pa} is shorter than the programmed value for t_{sa}.

21.14.2 Byte Access Type in Page Mode

The Byte Access Type configuration remains active in page mode. For 16-bit or 32-bit page mode devices that require byte selection signals, configure the BAT field of the SMC_REGISTER to 0 (byte select access type).

21.14.3 Page Mode Restriction

The page mode is not compatible with the use of the NWAIT signal. Using the page mode and the NWAIT signal may lead to unpredictable behavior.

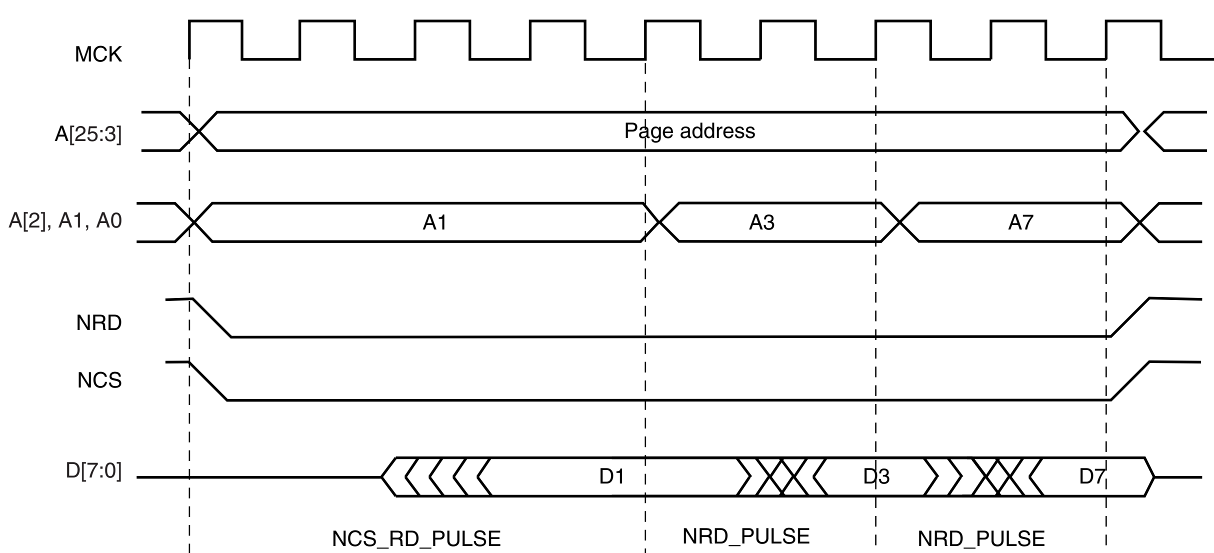
21.14.4 Sequential and Non-sequential Accesses

If the chip select and the MSB of addresses as defined in Table 21-7 are identical, then the current access lies in the same page as the previous one, and no page break occurs.

Using this information, all data within the same page, sequential or not sequential, are accessed with a minimum access time (t_{sa}). Figure 21-36 illustrates access to an 8-bit memory device in page mode, with 8-byte pages. Access to D1 causes a page access with a long access time (t_{pa}). Accesses to D3 and D7, though they are not sequential accesses, only require a short access time (t_{sa}).

If the MSB of addresses are different, the SMC performs the access of a new page. In the same way, if the chip select is different from the previous access, a page break occurs. If two sequential accesses are made to the page mode memory, but separated by an other internal or external peripheral access, a page break occurs on the second access because the chip select of the device was deasserted between both accesses.

Figure 21-36. Access to Non-sequential Data within the Same Page



21.15 Static Memory Controller (SMC) User Interface

The SMC is programmed using the registers listed in Table 21-9. For each chip select, a set of 4 registers is used to program the parameters of the external device connected on it. In Table 21-9, “CS_number” denotes the chip select number. 16 bytes (0x10) are required per chip select.

The user must complete writing the configuration by writing any one of the SMC_MODE registers.

Table 21-9. SMC Register Mapping

Offset	Register	Name	Access	Reset State
0x10 x CS_number + 0x00	SMC Setup Register	SMC_SETUP	Read/Write	0x00000000
0x10 x CS_number + 0x04	SMC Pulse Register	SMC_PULSE	Read/Write	0x01010101
0x10 x CS_number + 0x08	SMC Cycle Register	SMC_CYCLE	Read/Write	0x00010001
0x10 x CS_number + 0x0C	SMC Mode Register	SMC_MODE	Read/Write	0x10001000

21.15.1 SMC Setup Register

Register Name: SMC_SETUP[0 ..7]

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	NCS_RD_SETUP					

23	22	21	20	19	18	17	16
–	–	NRD_SETUP					

15	14	13	12	11	10	9	8
–	–	NCS_WR_SETUP					

7	6	5	4	3	2	1	0
–	–	NWE_SETUP					

- **NWE_SETUP: NWE Setup Length**

The NWE signal setup length is defined as:

NWE setup length = (128* NWE_SETUP[5] + NWE_SETUP[4:0]) clock cycles

- **NCS_WR_SETUP: NCS Setup Length in WRITE Access**

In write access, the NCS signal setup length is defined as:

NCS setup length = (128* NCS_WR_SETUP[5] + NCS_WR_SETUP[4:0]) clock cycles

- **NRD_SETUP: NRD Setup Length**

The NRD signal setup length is defined in clock cycles as:

NRD setup length = (128* NRD_SETUP[5] + NRD_SETUP[4:0]) clock cycles

- **NCS_RD_SETUP: NCS Setup Length in READ Access**

In read access, the NCS signal setup length is defined as:

NCS setup length = (128* NCS_RD_SETUP[5] + NCS_RD_SETUP[4:0]) clock cycles

21.15.2 SMC Pulse Register

Register Name: SMC_PULSE[0..7]

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	NCS_RD_PULSE						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	NRD_PULSE						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	NCS_WR_PULSE						
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	NWE_PULSE						

- **NWE_PULSE: NWE Pulse Length**

The NWE signal pulse length is defined as:

NWE pulse length = (256* NWE_PULSE[6] + NWE_PULSE[5:0]) clock cycles

The NWE pulse length must be at least 1 clock cycle.

- **NCS_WR_PULSE: NCS Pulse Length in WRITE Access**

In write access, the NCS signal pulse length is defined as:

NCS pulse length = (256* NCS_WR_PULSE[6] + NCS_WR_PULSE[5:0]) clock cycles

The NCS pulse length must be at least 1 clock cycle.

- **NRD_PULSE: NRD Pulse Length**

In standard read access, the NRD signal pulse length is defined in clock cycles as:

NRD pulse length = (256* NRD_PULSE[6] + NRD_PULSE[5:0]) clock cycles

The NRD pulse length must be at least 1 clock cycle.

In page mode read access, the NRD_PULSE parameter defines the duration of the subsequent accesses in the page.

- **NCS_RD_PULSE: NCS Pulse Length in READ Access**

In standard read access, the NCS signal pulse length is defined as:

NCS pulse length = (256* NCS_RD_PULSE[6] + NCS_RD_PULSE[5:0]) clock cycles

The NCS pulse length must be at least 1 clock cycle.

In page mode read access, the NCS_RD_PULSE parameter defines the duration of the first access to one page.

21.15.3 SMC Cycle Register

Register Name: SMC_CYCLE[0..7]

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	NRD_CYCLE

23	22	21	20	19	18	17	16
NRD_CYCLE							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	NWE_CYCLE

7	6	5	4	3	2	1	0
NWE_CYCLE							

- **NWE_CYCLE: Total Write Cycle Length**

The total write cycle length is the total duration in clock cycles of the write cycle. It is equal to the sum of the setup, pulse and hold steps of the NWE and NCS signals. It is defined as:

Write cycle length = (NWE_CYCLE[8:7]*256 + NWE_CYCLE[6:0]) clock cycles

- **NRD_CYCLE: Total Read Cycle Length**

The total read cycle length is the total duration in clock cycles of the read cycle. It is equal to the sum of the setup, pulse and hold steps of the NRD and NCS signals. It is defined as:

Read cycle length = (NRD_CYCLE[8:7]*256 + NRD_CYCLE[6:0]) clock cycles



21.15.4 SMC MODE Register

Register Name: SMC_MODE[0..7]

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	PS		–	–	–	PMEN

23	22	21	20	19	18	17	16
–	–	–	TDF_MODE	TDF_CYCLES			

15	14	13	12	11	10	9	8
–	–	DBW		–	–	–	BAT

7	6	5	4	3	2	1	0
–	–	EXNW_MODE		–	–	WRITE_MODE	READ_MODE

- **READ_MODE:**

1: The read operation is controlled by the NRD signal.

– If TDF cycles are programmed, the external bus is marked busy after the rising edge of NRD.

– If TDF optimization is enabled (TDF_MODE =1), TDF wait states are inserted after the setup of NRD.

0: The read operation is controlled by the NCS signal.

– If TDF cycles are programmed, the external bus is marked busy after the rising edge of NCS.

– If TDF optimization is enabled (TDF_MODE =1), TDF wait states are inserted after the setup of NCS.

- **WRITE_MODE**

1: The write operation is controlled by the NWE signal.

– If TDF optimization is enabled (TDF_MODE =1), TDF wait states will be inserted after the setup of NWE.

0: The write operation is controlled by the NCS signal.

– If TDF optimization is enabled (TDF_MODE =1), TDF wait states will be inserted after the setup of NCS.

- **EXNW_MODE: NWAIT Mode**

The NWAIT signal is used to extend the current read or write signal. It is only taken into account during the pulse phase of the read and write controlling signal. When the use of NWAIT is enabled, at least one cycle hold duration must be programmed for the read and write controlling signal.

EXNW_MODE		NWAIT Mode
0	0	Disabled
0	1	Reserved
1	0	Frozen Mode
1	1	Ready Mode

- Disabled Mode: The NWAIT input signal is ignored on the corresponding Chip Select.
- Frozen Mode: If asserted, the NWAIT signal freezes the current read or write cycle. After deassertion, the read/write cycle is resumed from the point where it was stopped.



- Ready Mode: The NWAIT signal indicates the availability of the external device at the end of the pulse of the controlling read or write signal, to complete the access. If high, the access normally completes. If low, the access is extended until NWAIT returns high.

• BAT: Byte Access Type

This field is used only if DBW defines a 16- or 32-bit data bus.

- 1: Byte write access type:
  – Write operation is controlled using NCS, NWR0, NWR1, NWR2, NWR3.
  – Read operation is controlled using NCS and NRD.
- 0: Byte select access type:
  – Write operation is controlled using NCS, NWE, NBS0, NBS1, NBS2 and NBS3
  – Read operation is controlled using NCS, NRD, NBS0, NBS1, NBS2 and NBS3

• DBW: Data Bus Width

DBW		Data Bus Width
0	0	8-bit bus
0	1	16-bit bus
1	0	32-bit bus
1	1	Reserved

• TDF_CYCLES: Data Float Time

This field gives the integer number of clock cycles required by the external device to release the data after the rising edge of the read controlling signal. The SMC always provide one full cycle of bus turnaround after the TDF_CYCLES period. The external bus cannot be used by another chip select during TDF_CYCLES + 1 cycles. From 0 up to 15 TDF_CYCLES can be set.

• TDF_MODE: TDF Optimization

1: TDF optimization is enabled.

– The number of TDF wait states is optimized using the setup period of the next read/write access.

0: TDF optimization is disabled.

– The number of TDF wait states is inserted before the next access begins.

• PMEN: Page Mode Enabled

1: Asynchronous burst read in page mode is applied on the corresponding chip select.

0: Standard read is applied.

• PS: Page Size

If page mode is enabled, this field indicates the size of the page in bytes.

PS		Page Size
0	0	4-byte page
0	1	8-byte page
1	0	16-byte page
1	1	32-byte page

22. SDRAM Controller (SDRAMC)

22.1 Description

The SDRAM Controller (SDRAMC) extends the memory capabilities of a chip by providing the interface to an external 16-bit or 32-bit SDRAM device. The page size supports ranges from 2048 to 8192 and the number of columns from 256 to 2048. It supports byte (8-bit), half-word (16-bit) and word (32-bit) accesses.

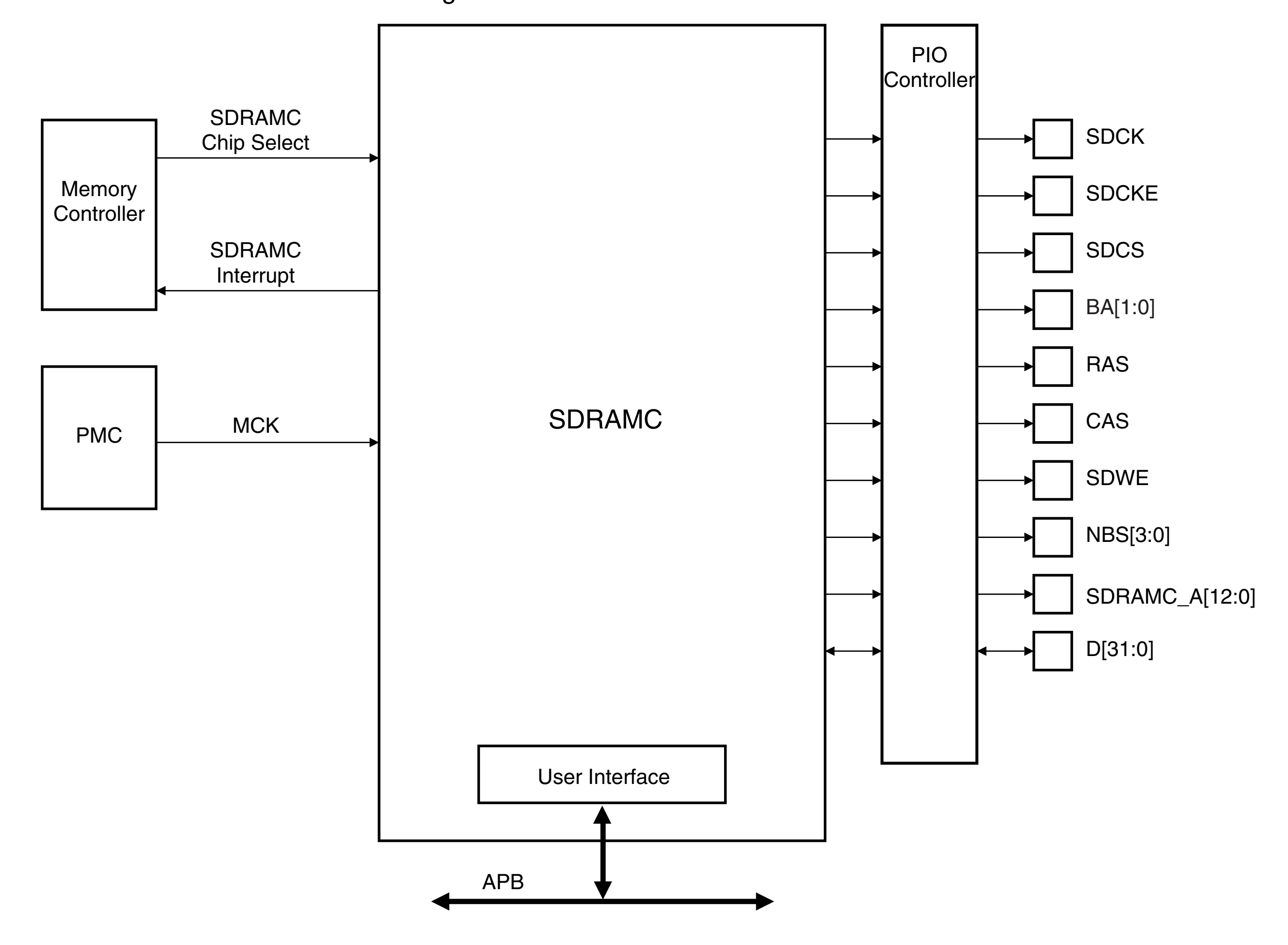
The SDRAM Controller supports a read or write burst length of one location. It keeps track of the active row in each bank, thus maximizing SDRAM performance, e.g., the application may be placed in one bank and data in the other banks. So as to optimize performance, it is advisable to avoid accessing different rows in the same bank.

The SDRAM controller supports a CAS latency of 1, 2 or 3 and optimizes the read access depending on the frequency.

The different modes available - self-refresh, power-down and deep power-down modes - minimize power consumption on the SDRAM device.

22.2 Block Diagram

Figure 22-1. SDRAM Controller Block Diagram





22.3 I/O Lines Description

Table 22-1. I/O Line Description

Name	Description	Type	Active Level
SDCK	SDRAM Clock	Output	
SDCKE	SDRAM Clock Enable	Output	High
SDCS	SDRAM Controller Chip Select	Output	Low
BA[1:0]	Bank Select Signals	Output	
RAS	Row Signal	Output	Low
CAS	Column Signal	Output	Low
SDWE	SDRAM Write Enable	Output	Low
NBS[3:0]	Data Mask Enable Signals	Output	Low
SDRAMC_A[12:0]	Address Bus	Output	
D[31:0]	Data Bus	I/O	

22.4 Application Example

22.4.1 Software Interface

The SDRAM address space is organized into banks, rows, and columns. The SDRAM controller allows mapping different memory types according to the values set in the SDRAMC configuration register.

The SDRAM Controller's function is to make the SDRAM device access protocol transparent to the user. Table 22-2 to Table 22-7 illustrate the SDRAM device memory mapping seen by the user in correlation with the device structure. Various configurations are illustrated.

22.4.1.1 32-bit Memory Data Bus Width

Table 22-2. SDRAM Configuration Mapping: 2K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Bk[1:0]		Row[10:0]											Column[7:0]								M[1:0]	
				Bk[1:0]		Row[10:0]											Column[8:0]									M[1:0]	
			Bk[1:0]		Row[10:0]											Column[9:0]										M[1:0]	
		Bk[1:0]		Row[10:0]											Column[10:0]											M[1:0]	

Table 22-3. SDRAM Configuration Mapping: 4K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Bk[1:0]		Row[11:0]												Column[7:0]								M[1:0]	
			Bk[1:0]		Row[11:0]												Column[8:0]									M[1:0]	
		Bk[1:0]		Row[11:0]												Column[9:0]										M[1:0]	
	Bk[1:0]		Row[11:0]												Column[10:0]											M[1:0]	

Table 22-4. SDRAM Configuration Mapping: 8K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
			Bk[1:0]		Row[12:0]													Column[7:0]								M[1:0]	
		Bk[1:0]		Row[12:0]													Column[8:0]									M[1:0]	
	Bk[1:0]		Row[12:0]													Column[9:0]										M[1:0]	
Bk[1:0]		Row[12:0]													Column[10:0]											M[1:0]	

Notes: 1. M[1:0] is the byte address inside a 32-bit word.
2. Bk[1] = BA1, Bk[0] = BA0.

22.4.1.2 16-bit Memory Data Bus Width

Table 22-5. SDRAM Configuration Mapping: 2K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
						Bk[1:0]		Row[10:0]											Column[7:0]								M0
					Bk[1:0]		Row[10:0]											Column[8:0]									M0
				Bk[1:0]		Row[10:0]											Column[9:0]										M0
			Bk[1:0]		Row[10:0]											Column[10:0]											M0

Table 22-6. SDRAM Configuration Mapping: 4K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
					Bk[1:0]		Row[11:0]												Column[7:0]								M0
				Bk[1:0]		Row[11:0]												Column[8:0]									M0
			Bk[1:0]		Row[11:0]												Column[9:0]										M0
		Bk[1:0]		Row[11:0]												Column[10:0]											M0

Table 22-7. SDRAM Configuration Mapping: 8K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
				Bk[1:0]		Row[12:0]													Column[7:0]								M0
			Bk[1:0]		Row[12:0]													Column[8:0]									M0
		Bk[1:0]		Row[12:0]													Column[9:0]										M0
	Bk[1:0]		Row[12:0]													Column[10:0]											M0

Notes: 1. M0 is the byte address inside a 16-bit half-word.
2. Bk[1] = BA1, Bk[0] = BA0.

22.5 Product Dependencies

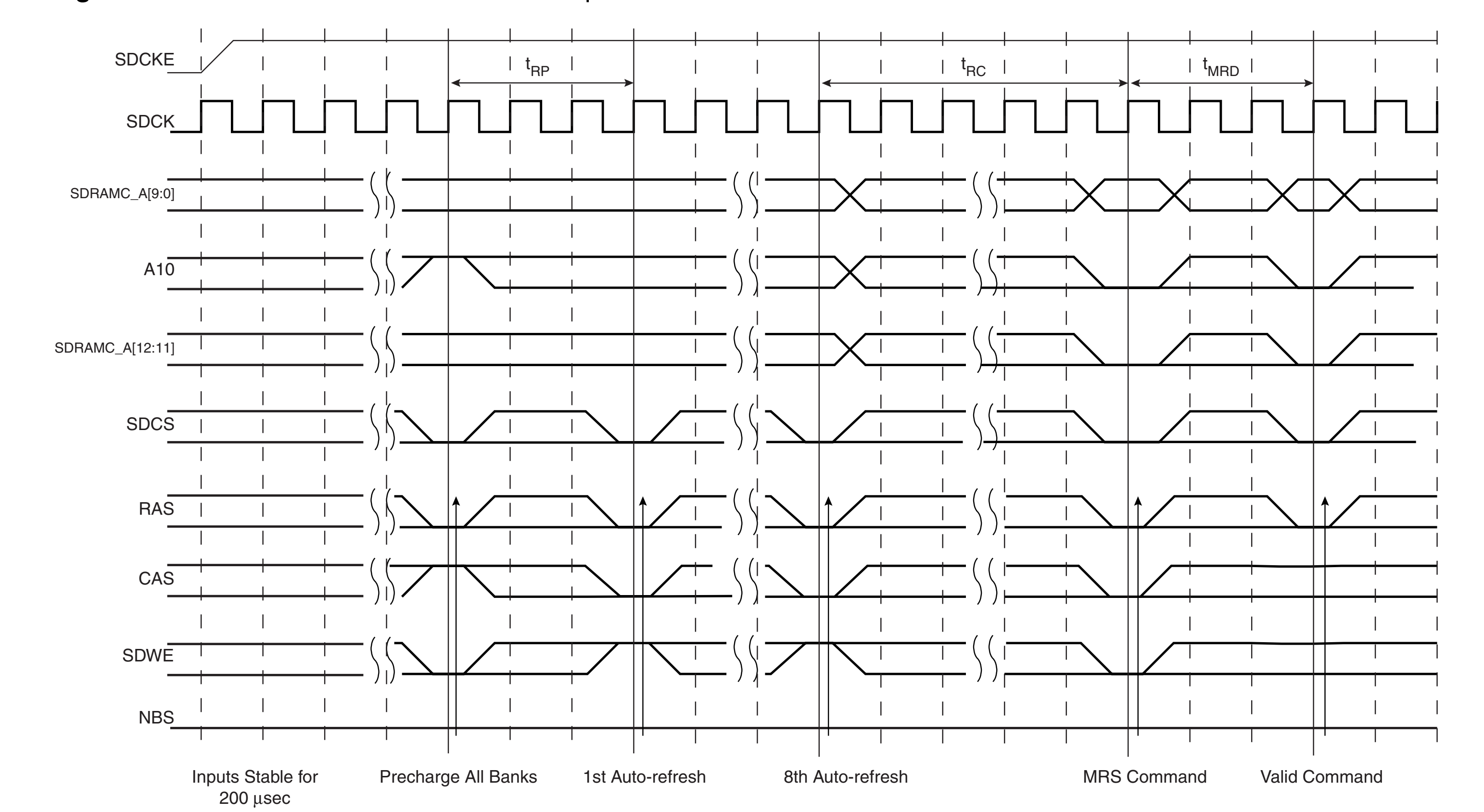
22.5.1 SDRAM Device Initialization

The initialization sequence is generated by software. The SDRAM devices are initialized by the following sequence:

1. SDRAM features must be set in the configuration register: asynchronous timings (TRC, TRAS, etc.), number of column, rows, CAS latency, and the data bus width.
2. For mobile SDRAM, temperature-compensated self refresh (TCSR), drive strength (DS) and partial array self refresh (PASR) must be set in the Low Power Register.
3. The SDRAM memory type must be set in the Memory Device Register.
4. A minimum pause of 200 µs is provided to precede any signal toggle.
5. An All Banks Precharge command is issued to the SDRAM devices. The application must set Mode to 2 in the Mode Register and perform a write access to any SDRAM address.
6. Eight auto-refresh (CBR) cycles are provided. The application must set the Mode to 4 in the Mode Register and performs a write access to any SDRAM location height times.
7. A Mode Register set (MRS) cycle is issued to program the parameters of the SDRAM devices, in particular CAS latency and burst length. The application must set Mode to 3 in the Mode Register and perform a write access to the SDRAM. The write address must be chosen so that BA[1:0] are set to 0. For example, with a 16-bit 128 MB SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be done at the address 0x20000000.
8. For mobile SDRAM initialization, an Extended Mode Register set (EMRS) cycle is issued to program the SDRAM parameters (TCSR, PASR, DS). The application must set Mode to 5 in the Mode Register and perform a write access to the SDRAM. The write address must be chosen so that BA[1] or BA[0] are set to 1. For example, with a 16-bit 128 MB SDRAM, (12 rows, 9 columns, 4 banks) bank address the SDRAM write access should be done at the address 0x20800000 or 0x20400000.
9. The application must go into Normal Mode, setting Mode to 0 in the Mode Register and performing a write access at any location in the SDRAM.
10. Write the refresh rate into the count field in the SDRAMC Refresh Timer register. (Refresh rate = delay between refresh cycles). The SDRAM device requires a refresh every 15.625 us or 7.81 us. With a 100 MHz frequency, the Refresh Timer Counter Register must be set with the value 1562(15.652 µs x 100 MHz) or 781(7.81 µs x 100 MHz).

After initialization, the SDRAM devices are fully functional.

Figure 22-2. SDRAM Device Initialization Sequence



22.5.2 I/O Lines

The pins used for interfacing the SDRAM Controller may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the SDRAM Controller pins to their peripheral function. If I/O lines of the SDRAM Controller are not used by the application, they can be used for other purposes by the PIO Controller.

22.5.3 Interrupt

The SDRAM Controller interrupt (Refresh Error notification) is connected to the Memory Controller. This interrupt may be ORed with other System Peripheral interrupt lines and is finally provided as the System Interrupt Source (Source 1) to the AIC (Advanced Interrupt Controller).

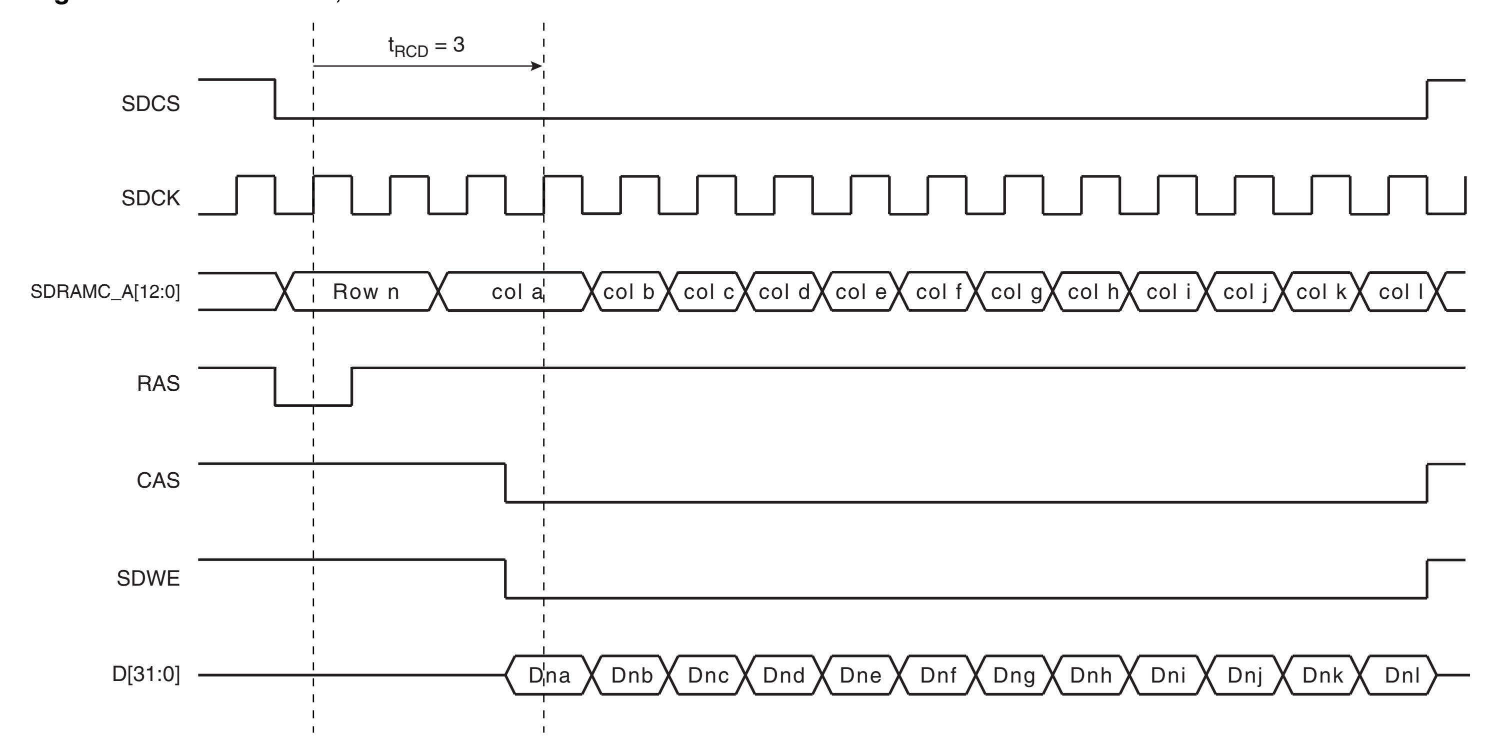
Using the SDRAM Controller interrupt requires the AIC to be programmed first.

22.6 Functional Description

22.6.1 SDRAM Controller Write Cycle

The SDRAM Controller allows burst access or single access. In both cases, the SDRAM controller keeps track of the active row in each bank, thus maximizing performance. To initiate a burst access, the SDRAM Controller uses the transfer type signal provided by the master requesting the access. If the next access is a sequential write access, writing to the SDRAM device is carried out. If the next access is a write-sequential access, but the current access is to a boundary page, or if the next access is in another row, then the SDRAM Controller generates a precharge command, activates the new row and initiates a write command. To comply with SDRAM timing parameters, additional clock cycles are inserted between precharge/active (t_{RP}) commands and active/write (t_{RCD}) commands. For definition of these timing parameters, refer to the ”SDRAMC Configuration Register” on page 202. This is described in Figure 22-3 below.

Figure 22-3. Write Burst, 32-bit SDRAM Access



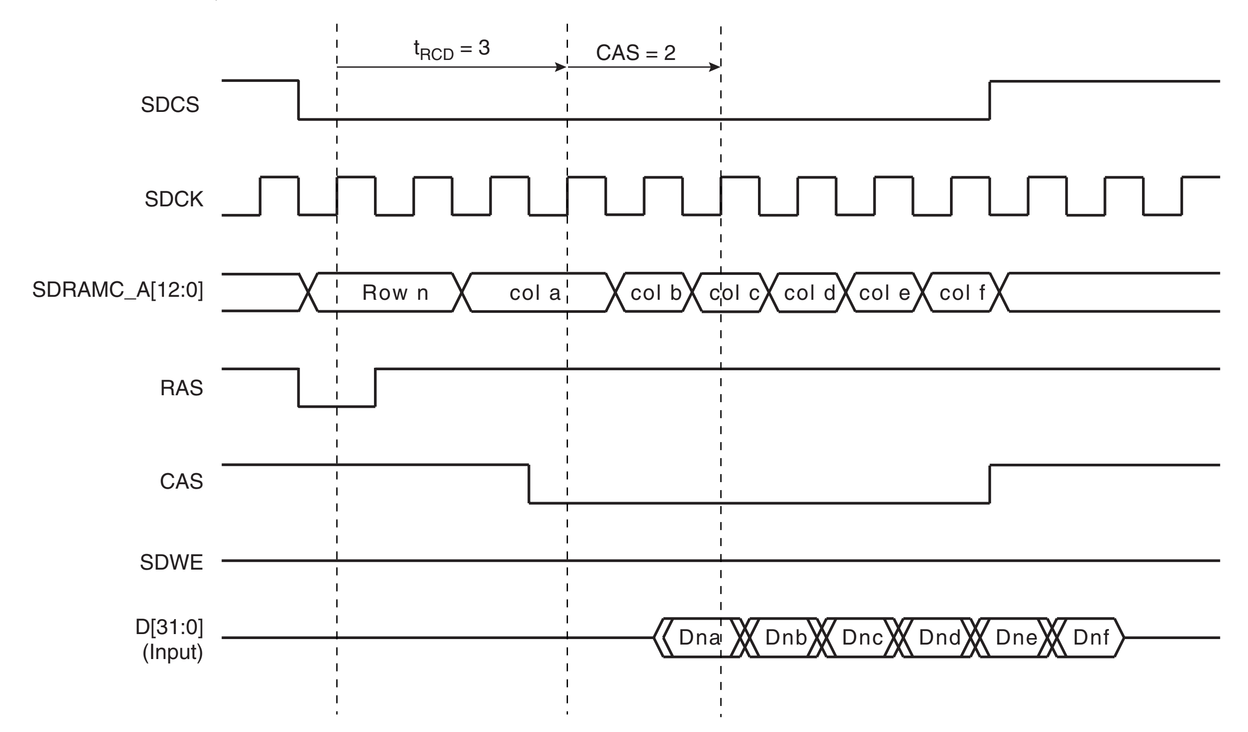
22.6.2 SDRAM Controller Read Cycle

The SDRAM Controller allows burst access, incremental burst of unspecified length or single access. In all cases, the SDRAM Controller keeps track of the active row in each bank, thus maximizing performance of the SDRAM. If row and bank addresses do not match the previous row/bank address, then the SDRAM controller automatically generates a precharge command, activates the new row and starts the read command. To comply with the SDRAM timing parameters, additional clock cycles on SDCK are inserted between precharge and active commands (t_{RP}) and between active and read command (t_{RCD}). These two parameters are set in the configuration register of the SDRAM Controller. After a read command, additional wait states are generated to comply with the CAS latency (1, 2 or 3 clock delays specified in the configuration register).

For a single access or an incremented burst of unspecified length, the SDRAM Controller anticipates the next access. While the last value of the column is returned by the SDRAM Controller on the bus, the SDRAM Controller anticipates the read to the next column and thus anticipates the CAS latency. This reduces the effect of the CAS latency on the internal bus.

For burst access of specified length (4, 8, 16 words), access is not anticipated. This case leads to the best performance. If the burst is broken (border, busy mode, etc.), the next access is handled as an incrementing burst of unspecified length.

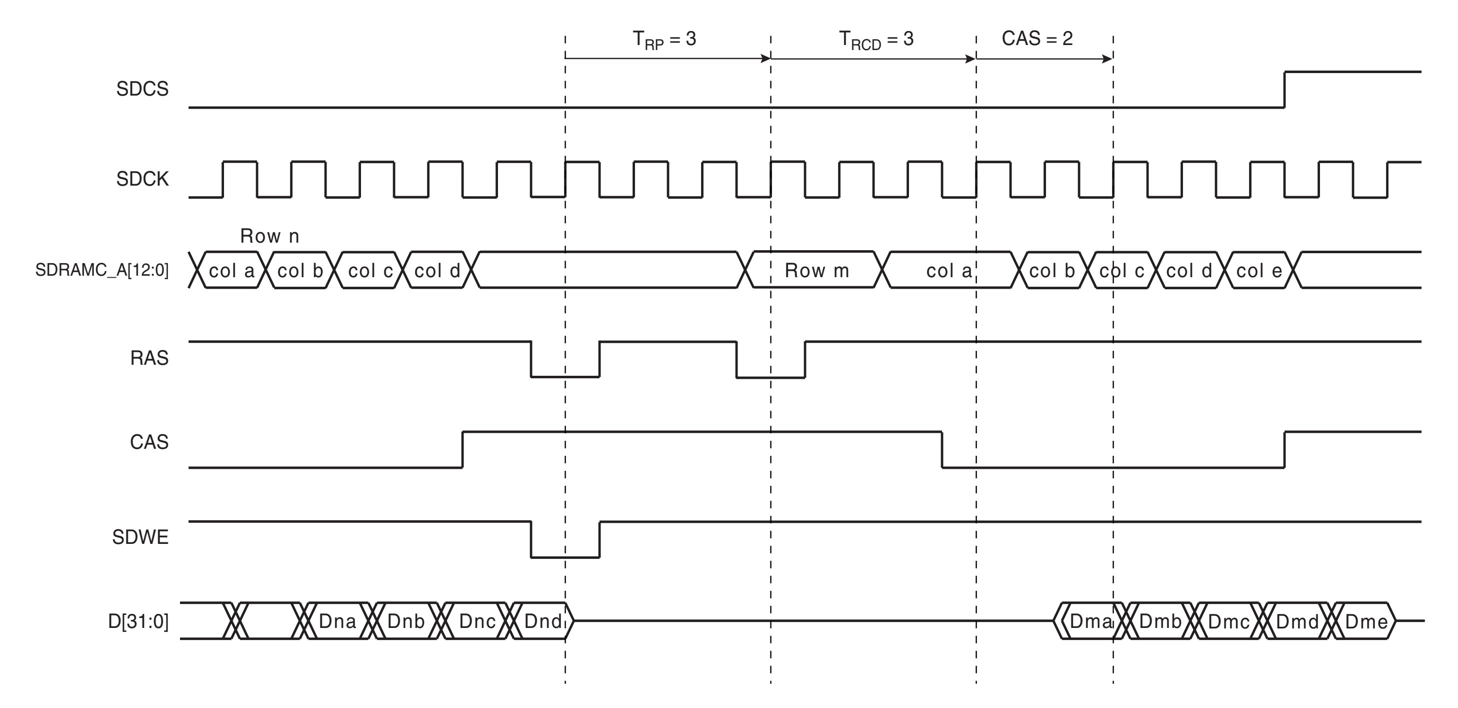
Figure 22-4. Read Burst, 32-bit SDRAM Access



22.6.3 Border Management

When the memory row boundary has been reached, an automatic page break is inserted. In this case, the SDRAM controller generates a precharge command, activates the new row and initiates a read or write command. To comply with SDRAM timing parameters, an additional clock cycle is inserted between the precharge/active (t_{RP}) command and the active/read (t_{RCD}) command. This is described in Figure 22-5 below.

Figure 22-5. Read Burst with Boundary Row Access



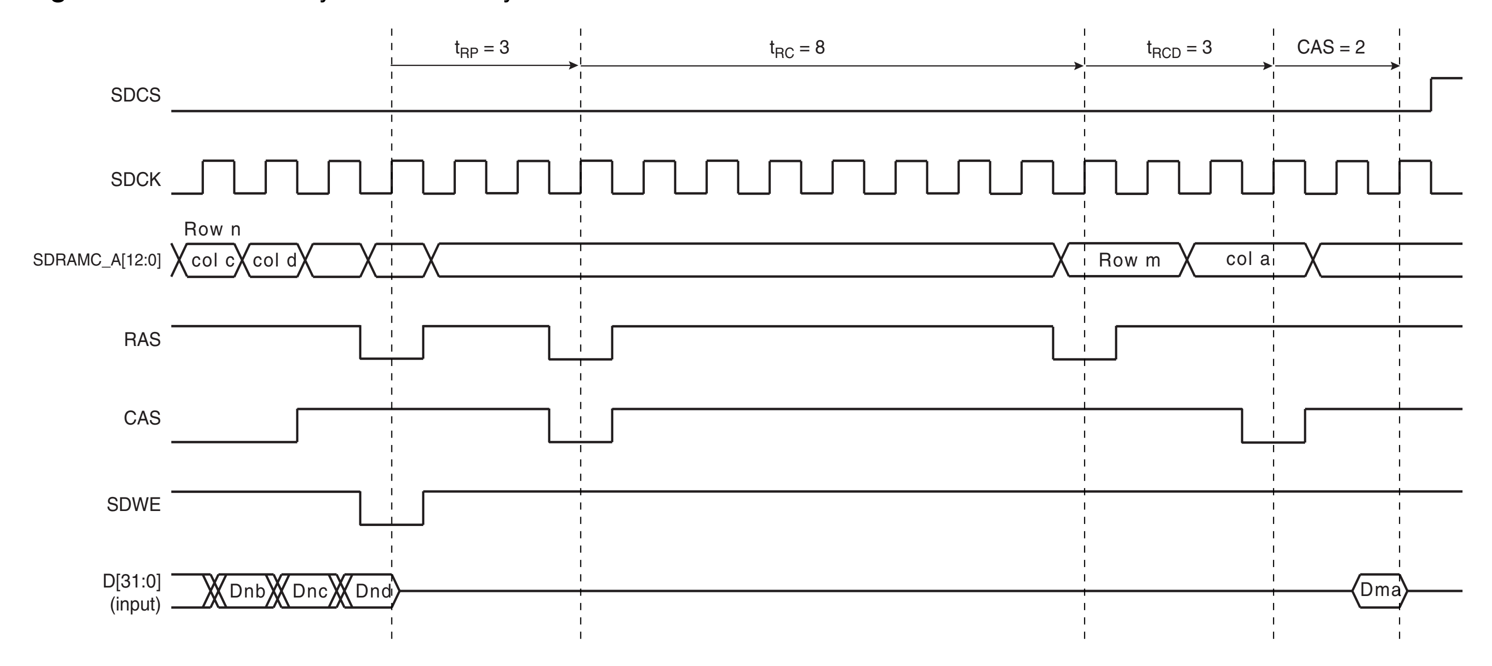
22.6.4 SDRAM Controller Refresh Cycles

An auto-refresh command is used to refresh the SDRAM device. Refresh addresses are generated internally by the SDRAM device and incremented after each auto-refresh automatically. The SDRAM Controller generates these auto-refresh commands periodically. An internal timer is loaded with the value in the register SDRAMC_TR that indicates the number of clock cycles between refresh cycles.

A refresh error interrupt is generated when the previous auto-refresh command did not perform. It is acknowledged by reading the Interrupt Status Register (SDRAMC_ISR).

When the SDRAM Controller initiates a refresh of the SDRAM device, internal memory accesses are not delayed. However, if the CPU tries to access the SDRAM, the slave indicates that the device is busy and the master is held by a wait signal. See Figure 22-6.

Figure 22-6. Refresh Cycle Followed by a Read Access



22.6.5 Power Management

Three low-power modes are available:

- Self-refresh Mode: The SDRAM executes its own Auto-refresh cycle without control of the SDRAM Controller. Current drained by the SDRAM is very low.
- Power-down Mode: Auto-refresh cycles are controlled by the SDRAM Controller. Between auto-refresh cycles, the SDRAM is in power-down. Current drained in Power-down mode is higher than in Self-refresh Mode.
- Deep Power-down Mode: (Only available with Mobile SDRAM) The SDRAM contents are lost, but the SDRAM does not drain any current.

The SDRAM Controller activates one low-power mode as soon as the SDRAM device is not selected. It is possible to delay the entry in self-refresh and power-down mode after the last access by programming a timeout value in the Low Power Register.

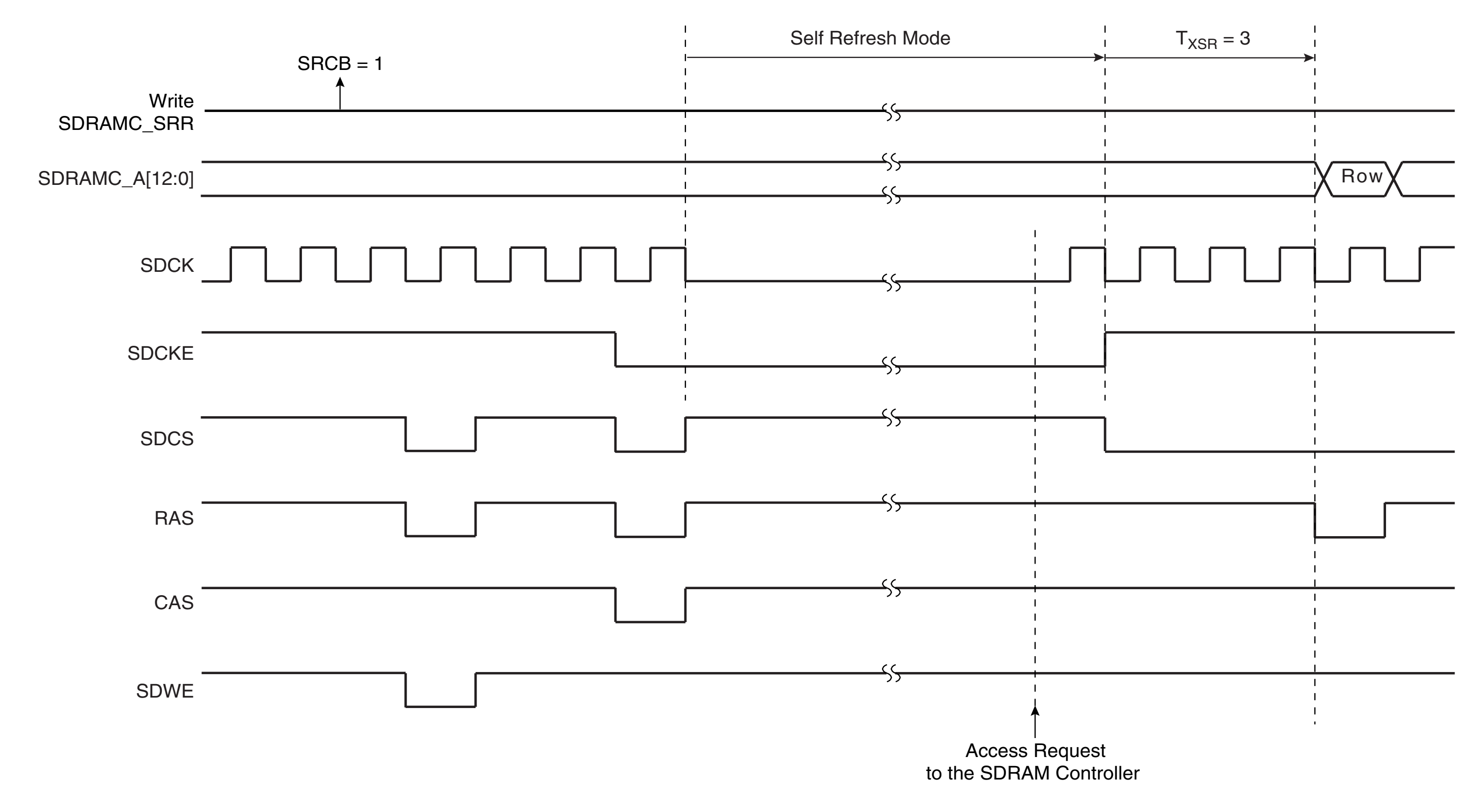
22.6.5.1 Self-refresh Mode

This mode is selected by programming the LPCB field to 1 in the SDRAMC Low Power Register. In self-refresh mode, the SDRAM device retains data without external clocking and provides its own internal clocking, thus performing its own auto-refresh cycles. All the inputs to the SDRAM device become “don’t care” except SDCKE, which remains low. As soon as the SDRAM device is selected, the SDRAM Controller provides a sequence of commands and exits self-refresh mode.

Some low-power SDRAMs (e.g., mobile SDRAM) can refresh only one-quarter or half quarter or all banks of the SDRAM array. This feature reduces the self refresh current. To configure this feature, Temperature Compensated Self Refresh (TCSR), Partial Array Self Refresh (PASR) and Drive Strength (DS) parameters must be set in the Low Power Register and transmitted to the low-power SDRAM during the initialization.

The SDRAM device must remain in self-refresh mode for a minimum period of t_{RAS} and may remain in self-refresh mode for an indefinite period. This is described in Figure 22-7.

Figure 22-7. Self-refresh Mode Behavior

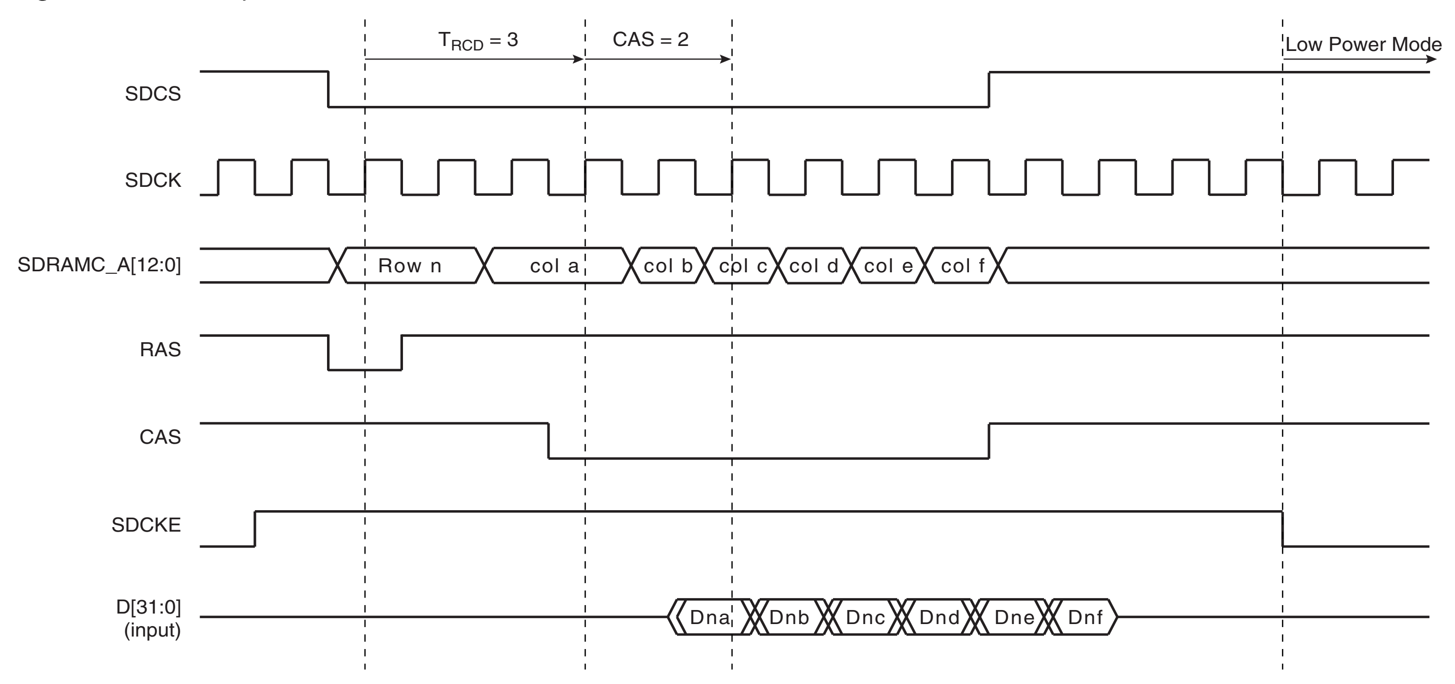


22.6.5.2 Low-power Mode

This mode is selected by programming the LPCB field to 2 in the SDRAMC Low Power Register. Power consumption is greater than in self-refresh mode. All the input and output buffers of the SDRAM device are deactivated except SDCKE, which remains low. In contrast to self-refresh mode, the SDRAM device cannot remain in low-power mode longer than the refresh period (64 ms for a whole device refresh operation). As no auto-refresh operations are performed by the SDRAM itself, the SDRAM Controller carries out the refresh operation. The exit procedure is faster than in self-refresh mode.

This is described in Figure 22-8.

Figure 22-8. Low-power Mode Behavior



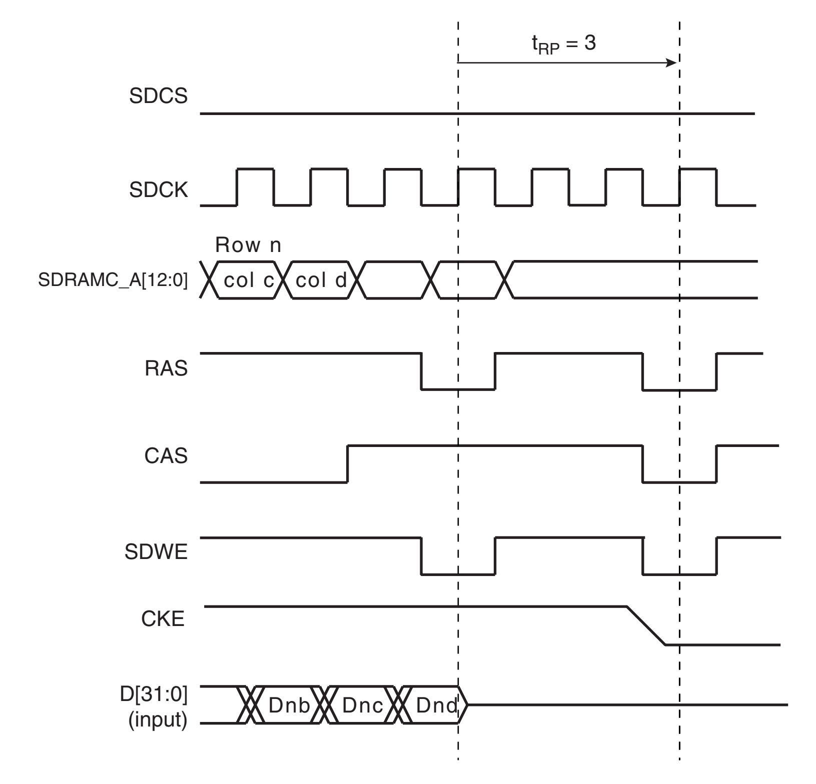
22.6.5.3 Deep Power-down Mode

This mode is selected by programming the LPCB field to 3 in the SDRAMC Low Power Register. When this mode is activated, all internal voltage generators inside the SDRAM are stopped and all data is lost.

When this mode is enabled, the application must not access to the SDRAM until a new initialization sequence is done (See "SDRAM Device Initialization" on page 190).

This is described in Figure 22-9.

Figure 22-9. Deep Power-down Mode Behavior



22.7 SDRAM Controller (SDRAMC) User Interface

Table 22-8. SDRAM Controller Memory Map

Offset	Register	Name	Access	Reset State
0x00	SDRAMC Mode Register	SDRAMC_MR	Read/Write	0x00000000
0x04	SDRAMC Refresh Timer Register	SDRAMC_TR	Read/Write	0x00000000
0x08	SDRAMC Configuration Register	SDRAMC_CR	Read/Write	0x852372C0
0x0C	SDRAMC High Speed Register	SDRAMC_HSR	Read/Write	0x00
0x10	SDRAMC Low Power Register	SDRAMC_LPR	Read/Write	0x0
0x14	SDRAMC Interrupt Enable Register	SDRAMC_IER	Write-only	–
0x18	SDRAMC Interrupt Disable Register	SDRAMC_IDR	Write-only	–
0x1C	SDRAMC Interrupt Mask Register	SDRAMC_IMR	Read-only	0x0
0x20	SDRAMC Interrupt Status Register	SDRAMC_ISR	Read-only	0x0
0x24	SDRAMC Memory Device Register	SDRAMC_MDR	Read	0x0
0x28 - 0xFC	Reserved	–	–	–

22.7.1 SDRAMC Mode Register

Register Name: SDRAMC_MR

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	MODE		

- **MODE: SDRAMC Command Mode**

This field defines the command issued by the SDRAM Controller when the SDRAM device is accessed.

MODE			Description
0	0	0	Normal mode. Any access to the SDRAM is decoded normally.
0	0	1	The SDRAM Controller issues a NOP command when the SDRAM device is accessed regardless of the cycle.
0	1	0	The SDRAM Controller issues an “All Banks Precharge” command when the SDRAM device is accessed regardless of the cycle.
0	1	1	The SDRAM Controller issues a “Load Mode Register” command when the SDRAM device is accessed regardless of the cycle. The address offset with respect to the SDRAM device base address is used to program the Mode Register. For instance, when this mode is activated, an access to the “SDRAM_Base + offset” address generates a “Load Mode Register” command with the value “offset” written to the SDRAM device Mode Register.
1	0	0	The SDRAM Controller issues an “Auto-Refresh” Command when the SDRAM device is accessed regardless of the cycle. Previously, an “All Banks Precharge” command must be issued.
1	0	1	The SDRAM Controller issues an extended load mode register command when the SDRAM device is accessed regardless of the cycle. The address offset with respect to the SDRAM device base address is used to program the Mode Register. For instance, when this mode is activated, an access to the “SDRAM_Base + offset” address generates an “Extended Load Mode Register” command with the value “offset” written to the SDRAM device Mode Register.
1	1	0	Deep power-down mode. Enters deep power-down mode.

22.7.2 SDRAMC Refresh Timer Register

Register Name: SDRAMC_TR

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	COUNT			

7	6	5	4	3	2	1	0
COUNT							

- **COUNT: SDRAMC Refresh Timer Count**

This 12-bit field is loaded into a timer that generates the refresh pulse. Each time the refresh pulse is generated, a refresh burst is initiated. The value to be loaded depends on the SDRAMC clock frequency (MCK: Master Clock), the refresh rate of the SDRAM device and the refresh burst length where 15.6 µs per row is a typical value for a burst of length one.

To refresh the SDRAM device, this 12-bit field must be written. If this condition is not satisfied, no refresh command is issued and no refresh of the SDRAM device is carried out.



22.7.3 SDRAMC Configuration Register

Register Name: SDRAMC_CR

Access Type: Read/Write

Reset Value: 0x852372C0

31	30	29	28	27	26	25	24
TXSR				TRAS			

23	22	21	20	19	18	17	16
TRCD				TRP			

15	14	13	12	11	10	9	8
TRC				TWR			

7	6	5	4	3	2	1	0
DBW	CAS		NB	NR		NC	

- **NC: Number of Column Bits**

Reset value is 8 column bits.

NC		Column Bits
0	0	8
0	1	9
1	0	10
1	1	11

- **NR: Number of Row Bits**

Reset value is 11 row bits.

NR		**Row Bits**
0	0	11
0	1	12
1	0	13
1	1	Reserved

- **NB: Number of Banks**

Reset value is two banks.

NB	**Number of Banks**
0	2
1	4

- **CAS: CAS Latency**

Reset value is two cycles.

In the SDRAMC, only a CAS latency of one, two and three cycles are managed. In any case, another value must be programmed.

CAS		CAS Latency (Cycles)
0	0	Reserved
0	1	1
1	0	2
1	1	3

- **DBW: Data Bus Width**

Reset value is 16 bits

0: Data bus width is 32 bits.

1: Data bus width is 16 bits.

- **TWR: Write Recovery Delay**

Reset value is two cycles.

This field defines the Write Recovery Time in number of cycles. Number of cycles is between 0 and 15.

- **TRC: Row Cycle Delay**

Reset value is seven cycles.

This field defines the delay between a Refresh and an Activate Command in number of cycles. Number of cycles is between 0 and 15.

- **TRP: Row Precharge Delay**

Reset value is three cycles.

This field defines the delay between a Precharge Command and another Command in number of cycles. Number of cycles is between 0 and 15.

- **TRCD: Row to Column Delay**

Reset value is two cycles.

This field defines the delay between an Activate Command and a Read/Write Command in number of cycles. Number of cycles is between 0 and 15.

- **TRAS: Active to Precharge Delay**

Reset value is five cycles.

This field defines the delay between an Activate Command and a Precharge Command in number of cycles. Number of cycles is between 0 and 15.

- **TXSR: Exit Self Refresh to Active Delay**

Reset value is height cycles.

This field defines the delay between SCKE set high and an Activate Command in number of cycles. Number of cycles is between 0 and 15.

22.7.4 SDRAMC High Speed Register

Register Name: SDRAMC_HSR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	DA

- **DA: Decode Cycle Enable**

0: Decode cycle is disabled.

1: Decode cycle is enabled.

22.7.5 SDRAMC Low Power Register

Register Name: SDRAMC_LPR

Access Type: Read/Write

Reset Value: 0x0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	TIMEOUT		DS		TCSR	

7	6	5	4	3	2	1	0
–	PASR			–	–	LPCB	

- **LPCB: Low-power Configuration Bits**

00	Low Power Feature is inhibited: no Power-down, Self-refresh or Deep Power-down command is issued to the SDRAM device.
01	The SDRAM Controller issues a Self-refresh command to the SDRAM device, the SDCLK clock is deactivated and the SDCKE signal is set low. The SDRAM device leaves the Self Refresh Mode when accessed and enters it after the access.
10	The SDRAM Controller issues a Power-down Command to the SDRAM device after each access, the SDCKE signal is set to low. The SDRAM device leaves the Power-down Mode when accessed and enters it after the access.
11	The SDRAM Controller issues a Deep Power-down command to the SDRAM device. This mode is unique to low-power SDRAM.

- **PASR: Partial Array Self-refresh (only for low-power SDRAM)**

PASR parameter is transmitted to the SDRAM during initialization to specify whether only one quarter, one half or all banks of the SDRAM array are enabled. Disabled banks are not refreshed in self-refresh mode. This parameter must be set according to the SDRAM device specification.

- **TCSR: Temperature Compensated Self-Refresh (only for low-power SDRAM)**

TCSR parameter is transmitted to the SDRAM during initialization to set the refresh interval during self-refresh mode depending on the temperature of the low-power SDRAM. This parameter must be set according to the SDRAM device specification.

- **DS: Drive Strength (only for low-power SDRAM)**

DS parameter is transmitted to the SDRAM during initialization to select the SDRAM strength of data output. This parameter must be set according to the SDRAM device specification.

- **TIMEOUT: Time to define when low-power mode is enabled**

00	The SDRAM controller activates the SDRAM low-power mode immediately after the end of the last transfer.
01	The SDRAM controller activates the SDRAM low-power mode 64 clock cycles after the end of the last transfer.
10	The SDRAM controller activates the SDRAM low-power mode 128 clock cycles after the end of the last transfer.
11	Reserved.

22.7.6 SDRAMC Interrupt Enable Register

Register Name: SDRAMC_IER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	RES

- **RES: Refresh Error Status**

0: No effect.

1: Enables the refresh error interrupt.

22.7.7 SDRAMC Interrupt Disable Register

Register Name: SDRAMC_IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	RES

- **RES: Refresh Error Status**

0: No effect.

1: Disables the refresh error interrupt.

22.7.8 SDRAMC Interrupt Mask Register

Register Name: SDRAMC_IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	RES

- **RES: Refresh Error Status**

0: The refresh error interrupt is disabled.

1: The refresh error interrupt is enabled.

22.7.9 SDRAMC Interrupt Status Register

Register Name: SDRAMC_ISR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	RES

- **RES: Refresh Error Status**

0: No refresh error has been detected since the register was last read.

1: A refresh error has been detected since the register was last read.



22.7.10 SDRAMC Memory Device Register

Register Name: SDRAMC_MDR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	MD	

- **MD: Memory Device Type**

00	SDRAM
01	Low-power SDRAM
10	Reserved
11	Reserved.

23. Peripheral DMA Controller (PDC)

23.1 Description

The Peripheral DMA Controller (PDC) transfers data between on-chip serial peripherals and the on- and/or off-chip memories. The link between the PDC and a serial peripheral is operated by the AHB to ABP bridge.

The PDC contains 19 channels. The full-duplex peripherals feature 18 mono directional channels used in pairs (transmit only or receive only). The half-duplex peripherals feature 1 bi-directional channels.

The user interface of each PDC channel is integrated into the user interface of the peripheral it serves. The user interface of mono directional channels (receive only or transmit only), contains two 32-bit memory pointers and two 16-bit counters, one set (pointer, counter) for current transfer and one set (pointer, counter) for next transfer. The bi-directional channel user interface contains four 32-bit memory pointers and four 16-bit counters. Each set (pointer, counter) is used by current transmit, next transmit, current receive and next receive.

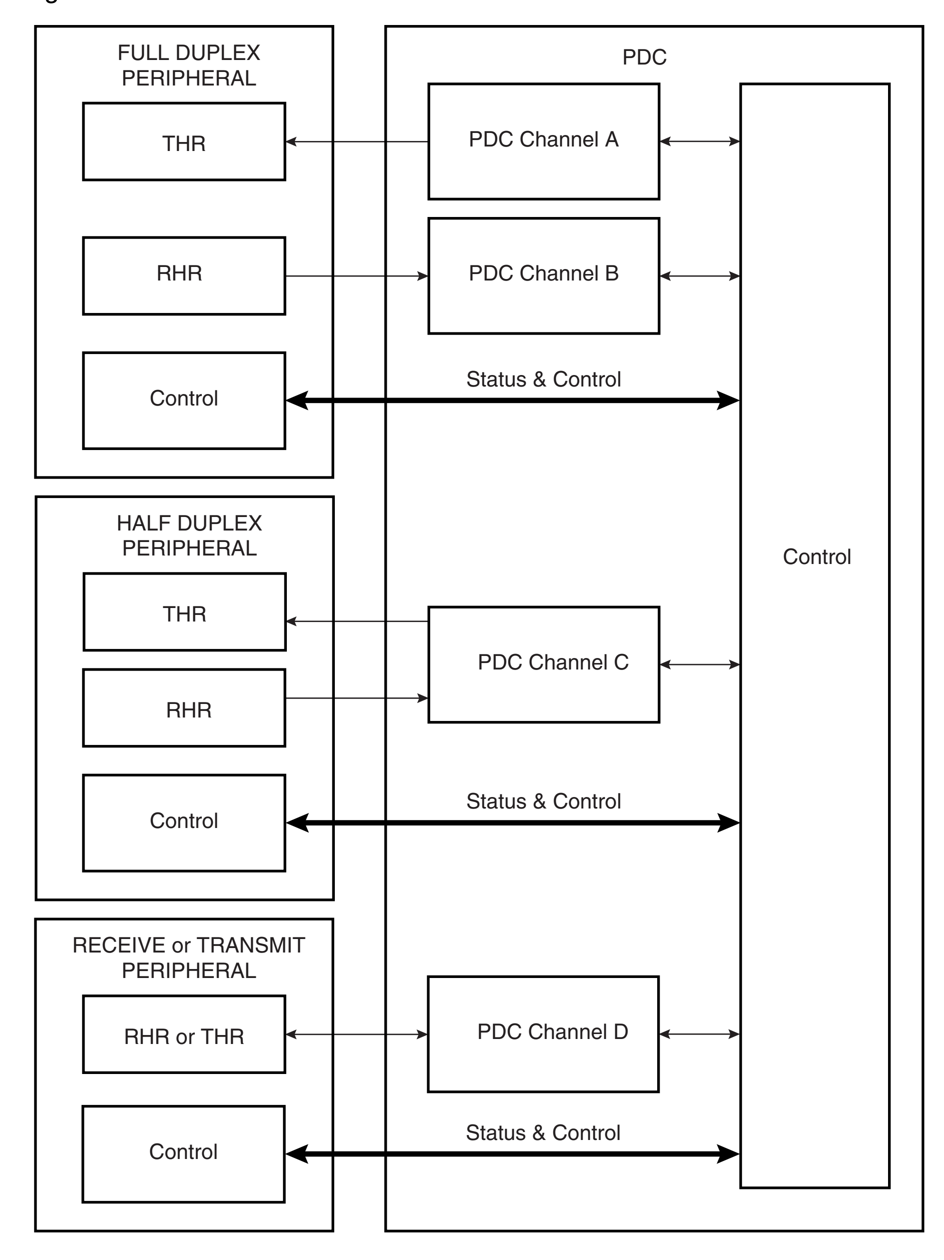
Using the PDC removes processor overhead by reducing its intervention during the transfer. This significantly reduces the number of clock cycles required for a data transfer, which improves microcontroller performance.

To launch a transfer, the peripheral triggers its associated PDC channels by using transmit and receive signals. When the programmed data is transferred, an end of transfer interrupt is generated by the peripheral itself.



23.2 Block Diagram

Figure 23-1. Block Diagram



23.3 Functional Description

23.3.1 Configuration

The PDC channel user interface enables the user to configure and control data transfers for each channel. The user interface of each PDC channel is integrated into the associated peripheral user interface.

The user interface of a serial peripheral, whether it is full or half duplex, contains four 32-bit pointers (RPR, RNPR, TPR, TNPR) and four 16-bit counter registers (RCR, RNCR, TCR, TNCR). However, the transmit and receive parts of each type are programmed differently: the transmit and receive parts of a full duplex peripheral can be programmed at the same time, whereas only one part (transmit or receive) of a half duplex peripheral can be programmed at a time.

32-bit pointers define the access location in memory for current and next transfer, whether it is for read (transmit) or write (receive). 16-bit counters define the size of current and next transfers. It is possible, at any moment, to read the number of transfers left for each channel.

The PDC has dedicated status registers which indicate if the transfer is enabled or disabled for each channel. The status for each channel is located in the associated peripheral status register. Transfers can be enabled and/or disabled by setting TXTEN/TXTDIS and RXTEN/RXTDIS in the peripheral's Transfer Control Register.

At the end of a transfer, the PDC channel sends status flags to its associated peripheral. These flags are visible in the peripheral status register (ENDRX, ENDTX, RXBUFF, and TXBUFE). Refer to Section 23.3.3 and to the associated peripheral user interface.

23.3.2 Memory Pointers

Each full duplex peripheral is connected to the PDC by a receive channel and a transmit channel. Both channels have 32-bit memory pointers that point respectively to a receive area and to a transmit area in on- and/or off-chip memory.

Each half duplex peripheral is connected to the PDC by a bidirectional channel. This channel has two 32-bit memory pointers, one for current transfer and the other for next transfer. These pointers point to transmit or receive data depending on the operating mode of the peripheral.

Depending on the type of transfer (byte, half-word or word), the memory pointer is incremented respectively by 1, 2 or 4 bytes.

If a memory pointer address changes in the middle of a transfer, the PDC channel continues operating using the new address.

23.3.3 Transfer Counters

Each channel has two 16-bit counters, one for current transfer and the other one for next transfer. These counters define the size of data to be transferred by the channel. The current transfer counter is decremented first as the data addressed by current memory pointer starts to be transferred. When the current transfer counter reaches zero, the channel checks its next transfer counter. If the value of next counter is zero, the channel stops transferring data and sets the appropriate flag. But if the next counter value is greater then zero, the values of the next pointer/next counter are copied into the current pointer/current counter and the channel resumes the transfer whereas next pointer/next counter get zero/zero as values. At the end of this transfer the PDC channel sets the appropriate flags in the Peripheral Status Register.

The following list gives an overview of how status register flags behave depending on the counters' values:

- ENDRX flag is set when the PERIPH_RCR register reaches zero.
- RXBUFF flag is set when both PERIPH_RCR and PERIPH_RNCR reach zero.
- ENDTX flag is set when the PERIPH_TCR register reaches zero.
- TXBUFE flag is set when both PERIPH_TCR and PERIPH_TNCR reach zero.

These status flags are described in the Peripheral Status Register.

23.3.4 Data Transfers

The serial peripheral triggers its associated PDC channels' transfers using transmit enable (TXEN) and receive enable (RXEN) flags in the transfer control register integrated in the peripheral's user interface.

When the peripheral receives an external data, it sends a Receive Ready signal to its PDC receive channel which then requests access to the Matrix. When access is granted, the PDC receive channel starts reading the peripheral Receive Holding Register (RHR). The read data are stored in an internal buffer and then written to memory.

When the peripheral is about to send data, it sends a Transmit Ready to its PDC transmit channel which then requests access to the Matrix. When access is granted, the PDC transmit channel reads data from memory and puts them to Transmit Holding Register (THR) of its associated peripheral. The same peripheral sends data according to its mechanism.

23.3.5 PDC Flags and Peripheral Status Register

Each peripheral connected to the PDC sends out receive ready and transmit ready flags and the PDC sends back flags to the peripheral. All these flags are only visible in the Peripheral Status Register.

Depending on the type of peripheral, half or full duplex, the flags belong to either one single channel or two different channels.

23.3.5.1 Receive Transfer End

This flag is set when PERIPH_RCR register reaches zero and the last data has been transferred to memory.

It is reset by writing a non zero value in PERIPH_RCR or PERIPH_RNCR.

23.3.5.2 Transmit Transfer End

This flag is set when PERIPH_TCR register reaches zero and the last data has been written into peripheral THR.

It is reset by writing a non zero value in PERIPH_TCR or PERIPH_TNCR.

23.3.5.3 Receive Buffer Full

This flag is set when PERIPH_RCR register reaches zero with PERIPH_RNCR also set to zero and the last data has been transferred to memory.

It is reset by writing a non zero value in PERIPH_TCR or PERIPH_TNCR.

23.3.5.4 Transmit Buffer Empty

This flag is set when PERIPH_TCR register reaches zero with PERIPH_TNCR also set to zero and the last data has been written into peripheral THR.

It is reset by writing a non zero value in PERIPH_TCR or PERIPH_TNCR.

23.4 Peripheral DMA Controller (PDC) User Interface

Table 23-1. Memory Map

Offset	Register	Name	Access	Reset State
0x100	Receive Pointer Register	PERIPH[(1)]_RPR	Read/Write	0
0x104	Receive Counter Register	PERIPH_RCR	Read/Write	0
0x108	Transmit Pointer Register	PERIPH_TPR	Read/Write	0
0x10C	Transmit Counter Register	PERIPH_TCR	Read/Write	0
0x110	Receive Next Pointer Register	PERIPH_RNPR	Read/Write	0
0x114	Receive Next Counter Register	PERIPH_RNCR	Read/Write	0
0x118	Transmit Next Pointer Register	PERIPH_TNPR	Read/Write	0
0x11C	Transmit Next Counter Register	PERIPH_TNCR	Read/Write	0
0x120	Transfer Control Register	PERIPH_PTCR	Write	0
0x124	Transfer Status Register	PERIPH_PTSR	Read	0

Note: 1. PERIPH: Ten registers are mapped in the peripheral memory space at the same offset. These can be defined by the user according to the function and the peripheral desired (DBGU, USART, SSC, SPI, MCI, etc.)

23.4.1 Receive Pointer Register

Register Name: PERIPH_RPR

Access Type: Read/Write

31	30	29	28	27	26	25	24
RXPTR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RXPTR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RXPTR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXPTR							

- **RXPTR: Receive Pointer Register**

RXPTR must be set to receive buffer address.

When a half duplex peripheral is connected to the PDC, RXPTR = TXPTR.

23.4.2 Receive Counter Register

Register Name: PERIPH_RCR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RXCTR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXCTR							

- **RXCTR: Receive Counter Register**

RXCTR must be set to receive buffer size.

When a half duplex peripheral is connected to the PDC, RXCTR = TXCTR.

0 = Stops peripheral data transfer to the receiver

1 - 65535 = Starts peripheral data transfer if corresponding channel is active

23.4.3 Transmit Pointer Register

Register Name: PERIPH_TPR

Access Type: Read/Write

31	30	29	28	27	26	25	24
TXPTR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TXPTR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXPTR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXPTR							

- **TXPTR: Transmit Counter Register**

TXPTR must be set to transmit buffer address.

When a half duplex peripheral is connected to the PDC, RXPTR = TXPTR.

23.4.4 Transmit Counter Register

Register Name: PERIPH_TCR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXCTR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXCTR							

- **TXCTR: Transmit Counter Register**

TXCTR must be set to transmit buffer size.

When a half duplex peripheral is connected to the PDC, RXCTR = TXCTR.

0 = Stops peripheral data transfer to the transmitter

1- 65535 = Starts peripheral data transfer if corresponding channel is active

23.4.5 Receive Next Pointer Register

Register Name: PERIPH_RNPR

Access Type: Read/Write

31	30	29	28	27	26	25	24
RXNPTR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RXNPTR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RXNPTR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXNPTR							

- **RXNPTR: Receive Next Pointer**

RXNPTR contains next receive buffer address.

When a half duplex peripheral is connected to the PDC, RXNPTR = TXNPTR.

23.4.6 Receive Next Counter Register

Register Name: PERIPH_RNCR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RXNCTR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXNCTR							

- **RXNCTR: Receive Next Counter**

RXNCTR contains next receive buffer size.

When a half duplex peripheral is connected to the PDC, RXNCTR = TXNCTR.

23.4.7 Transmit Next Pointer Register

Register Name: PERIPH_TNPR

Access Type: Read/Write

31	30	29	28	27	26	25	24
TXNPTR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TXNPTR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXNPTR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXNPTR							

- **TXNPTR: Transmit Next Pointer**

TXNPTR contains next transmit buffer address.

When a half duplex peripheral is connected to the PDC, RXNPTR = TXNPTR.

23.4.8 Transmit Next Counter Register

Register Name: PERIPH_TNCR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXNCTR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXNCTR							

- **TXNCTR: Transmit Counter Next**

TXNCTR contains next transmit buffer size.

When a half duplex peripheral is connected to the PDC, RXNCTR = TXNCTR.



23.4.9 Transfer Control Register

Register Name: PERIPH_PTCR

Access Type: Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	TXTDIS	TXTEN
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	RXTDIS	RXTEN

- **RXTEN: Receiver Transfer Enable**

0 = No effect.

1 = Enables PDC receiver channel requests if RXTDIS is not set.

When a half duplex peripheral is connected to the PDC, enabling the receiver channel requests automatically disables the transmitter channel requests. It is forbidden to set both TXTEN and RXTEN for a half duplex peripheral.

- **RXTDIS: Receiver Transfer Disable**

0 = No effect.

1 = Disables the PDC receiver channel requests.

When a half duplex peripheral is connected to the PDC, disabling the receiver channel requests also disables the transmitter channel requests.

- **TXTEN: Transmitter Transfer Enable**

0 = No effect.

1 = Enables the PDC transmitter channel requests.

When a half duplex peripheral is connected to the PDC, it enables the transmitter channel requests only if RXTEN is not set. It is forbidden to set both TXTEN and RXTEN for a half duplex peripheral.

- **TXTDIS: Transmitter Transfer Disable**

0 = No effect.

1 = Disables the PDC transmitter channel requests.

When a half duplex peripheral is connected to the PDC, disabling the transmitter channel requests disables the receiver channel requests.



23.4.10 Transfer Status Register

Register Name: PERIPH_PTSR

Access Type: Read

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	TXTEN
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	RXTEN

- **RXTEN: Receiver Transfer Enable**

0 = PDC Receiver channel requests are disabled.

1 = PDC Receiver channel requests are enabled.

- **TXTEN: Transmitter Transfer Enable**

0 = PDC Transmitter channel requests are disabled.

1 = PDC Transmitter channel requests are enabled.

24. Clock Generator

24.1 Description

The Clock Generator is made up of 2 PLLs, a Main Oscillator, and a 32,768 Hz low-power Oscillator.

It provides the following clocks:

- SLCK, the Slow Clock, is the only permanent clock within the system
- MAINCK is the output of the Main Oscillator

The Clock Generator User Interface is embedded within the Power Management Controller one and is described in Section 25.11. However, the Clock Generator registers are named CKGR_.

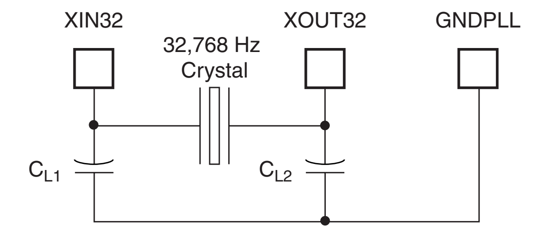
- PLLACK is the output of the Divider and PLL A block
- PLLBCK is the output of the Divider and PLL B block

24.2 Slow Clock Crystal Oscillator

The slow clock is the output of the slow clock generator and is the only clock considered permanent in a system that includes the Power Management Controller. It is mandatory in the operations of the PMC. In all cases, a 32,768 Hz crystal must be connected to the XIN32 and XOUT32 pins in order to ensure that the slow clock is present.

The Clock Generator integrates a 32,768 Hz low-power oscillator. The XIN32 and XOUT32 pins must be connected to a 32,768 Hz crystal. Two external capacitors must be wired as shown in Figure 24-1.

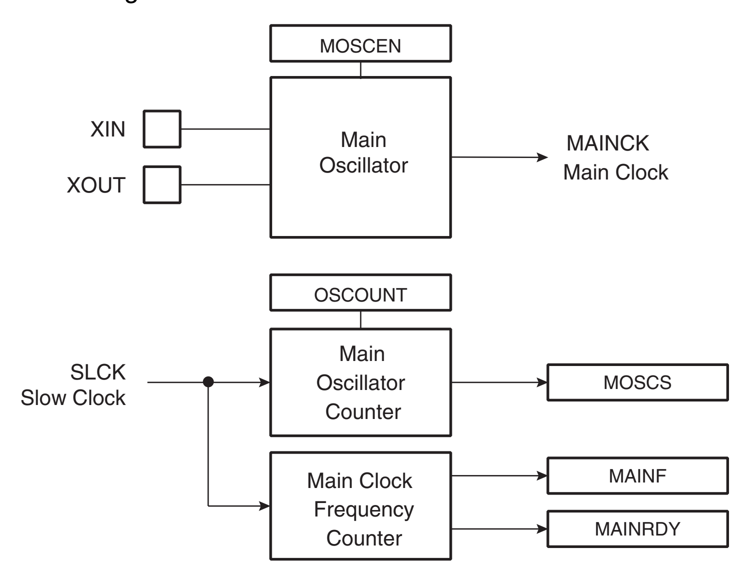
Figure 24-1. Typical Slow Clock Crystal Oscillator Connection



24.3 Main Oscillator

Figure 24-2 shows the Main Oscillator block diagram.

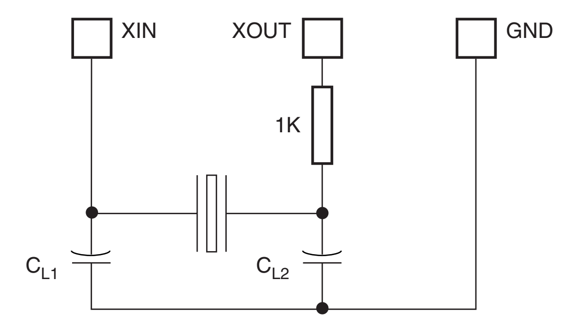
Figure 24-2. Main Oscillator Block Diagram



24.3.1 Main Oscillator Connections

The Clock Generator integrates a Main Oscillator that is designed for a 3 to 20 MHz fundamental crystal. The typical crystal connection is illustrated in Figure 24-3. The 1 kΩ resistor is only required for crystals with frequencies lower than 8 MHz. The oscillator contains 25 pF capacitors on each XIN and XOUT pin. Consequently, CL1 and CL2 can be removed when a crystal with a load capacitance of 12.5 pF is used. For further details on the electrical characteristics of the Main Oscillator, see the DC Characteristics section of the product datasheet.

Figure 24-3. Typical Crystal Connection



24.3.2 Main Oscillator Startup Time

The startup time of the Main Oscillator is given in the DC Characteristics section of the product datasheet. The startup time depends on the crystal frequency and decreases when the frequency rises.

24.3.3 Main Oscillator Control

To minimize the power required to start up the system, the main oscillator is disabled after reset and slow clock is selected.

The software enables or disables the main oscillator so as to reduce power consumption by clearing the MOSCEN bit in the Main Oscillator Register (CKGR_MOR).

When disabling the main oscillator by clearing the MOSCEN bit in CKGR_MOR, the MOSCS bit in PMC_SR is automatically cleared, indicating the main clock is off.

When enabling the main oscillator, the user must initiate the main oscillator counter with a value corresponding to the startup time of the oscillator. This startup time depends on the crystal frequency connected to the main oscillator.

When the MOSCEN bit and the OSCOUNT are written in CKGR_MOR to enable the main oscillator, the MOSCS bit in PMC_SR (Status Register) is cleared and the counter starts counting down on the slow clock divided by 8 from the OSCOUNT value. Since the OSCOUNT value is coded with 8 bits, the maximum startup time is about 62 ms.

When the counter reaches 0, the MOSCS bit is set, indicating that the main clock is valid. Setting the MOSCS bit in PMC_IMR can trigger an interrupt to the processor.

24.3.4 Main Clock Frequency Counter

The Main Oscillator features a Main Clock frequency counter that provides the quartz frequency connected to the Main Oscillator. Generally, this value is known by the system designer; however, it is useful for the boot program to configure the device with the correct clock speed, independently of the application.

The Main Clock frequency counter starts incrementing at the Main Clock speed after the next rising edge of the Slow Clock as soon as the Main Oscillator is stable, i.e., as soon as the MOSCS bit is set. Then, at the 16th falling edge of Slow Clock, the MAINRDY bit in CKGR_MCFR (Main Clock Frequency Register) is set and the counter stops counting. Its value can be read in the MAINF field of CKGR_MCFR and gives the number of Main Clock cycles during 16 periods of Slow Clock, so that the frequency of the crystal connected on the Main Oscillator can be determined.

24.3.5 Main Oscillator Bypass

The user can input a clock on the device instead of connecting a crystal. In this case, the user has to provide the external clock signal on the XIN pin. The input characteristics of the XIN pin under these conditions are given in the product electrical characteristics section. The programmer has to be sure to set the OSCBYPASS bit to 1 and the MOSCEN bit to 0 in the Main OSC register (CKGR_MOR) for the external clock to operate properly.

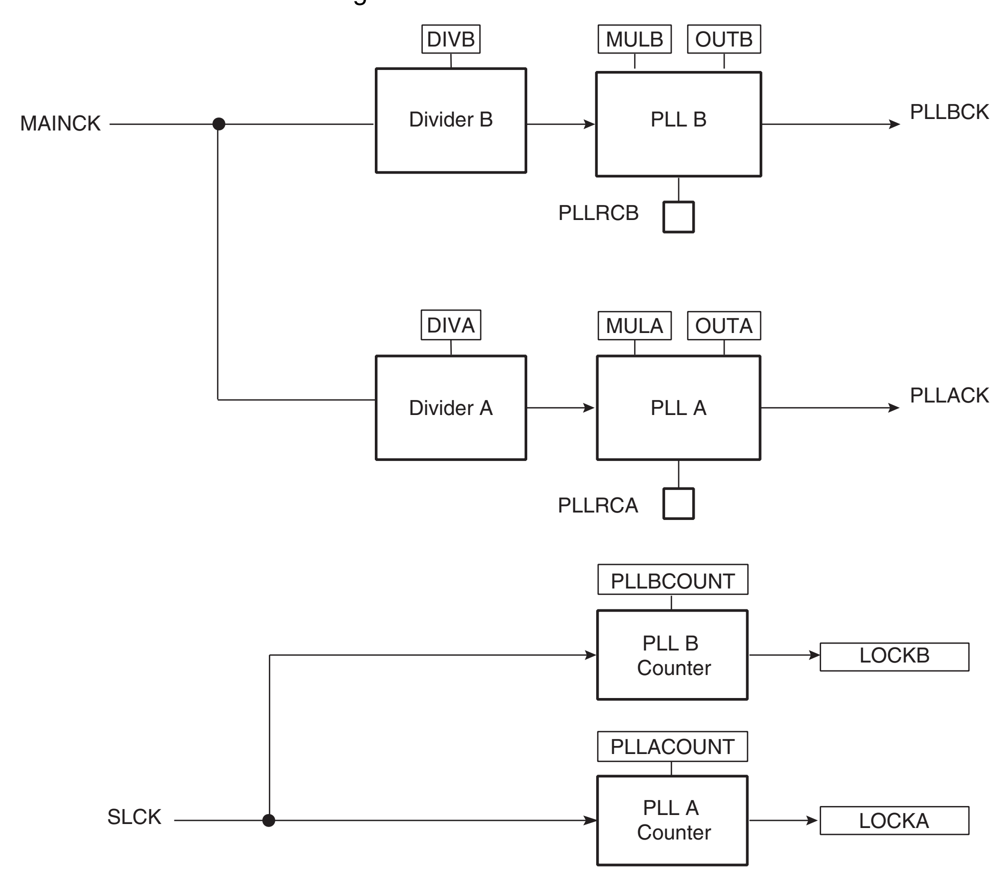


24.4 Divider and PLL Block

The PLL embeds an input divider to increase the accuracy of the resulting clock signals. However, the user must respect the PLL minimum input frequency when programming the divider.

Figure 24-4 shows the block diagram of the divider and PLL blocks.

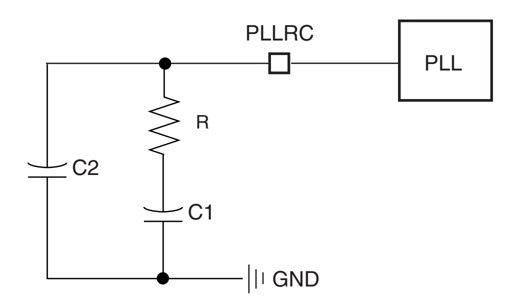
Figure 24-4. Divider and PLL Blocks Block Diagram



24.4.1 PLL Filter

The PLL requires connection to an external second-order filter through the pin, either PPLRCA or PLLRCB. Figure 24-6 shows a schematic of these filters.

Figure 24-5. PLL Capacitors and Resistors



Values of R, C1 and C2 to be connected to the PLLRC pin must be calculated as a function of the PLL input frequency, the PLL output frequency and the phase margin. A trade-off has to be found between output signal overshoot and startup time.



24.4.2 Divider and Phase Lock Loop Programming

The divider can be set between 1 and 255 in steps of 1. When a divider field (DIV) is set to 0, the output of the corresponding divider and the PLL output is a continuous signal at level 0. On reset, each DIV field is set to 0, thus the corresponding PLL input clock is set to 0.

The PLL allows multiplication of the divider's outputs. The PLL clock signal has a frequency that depends on the respective source signal frequency and on the parameters DIV and MUL. The factor applied to the source signal frequency is (MUL + 1)/DIV. When MUL is written to 0, the corresponding PLL is disabled and its power consumption is saved. Re-enabling the PLL can be performed by writing a value higher than 0 in the MUL field.

Whenever the PLL is re-enabled or one of its parameters is changed, the LOCK bit (LOCKA or LOCKB) in PMC_SR is automatically cleared. The values written in the PLLCOUNT field (PLLACOUNT or PLLBCOUNT) in CKGR_PLLR (CKGR_PLLAR or CKGR_PLLBR), are loaded in the PLL counter. The PLL counter then decrements at the speed of the Slow Clock until it reaches 0. At this time, the LOCK bit is set in PMC_SR and can trigger an interrupt to the processor. The user has to load the number of Slow Clock cycles required to cover the PLL transient time into the PLLCOUNT field. The transient time depends on the PLL filter. The initial state of the PLL and its target frequency can be calculated using a specific tool provided by Atmel.

25. Power Management Controller (PMC)

25.1 Description

The Power Management Controller (PMC) optimizes power consumption by controlling all system and user peripheral clocks. The PMC enables/disables the clock inputs to many of the peripherals and the ARM Processor.

The Power Management Controller provides the following clocks:

- MCK, the Master Clock, programmable from a few hundred Hz to the maximum operating frequency of the device. It is available to the modules running permanently, such as the AIC and the Memory Controller.
- Processor Clock (PCK), switched off when entering processor in idle mode.
- Peripheral Clocks, typically MCK, provided to the embedded peripherals (USART, SSC, SPI, TWI, TC, MCI, etc.) and independently controllable. In order to reduce the number of clock names in a product, the Peripheral Clocks are named MCK in the product datasheet.
- HClocks (MCK), provided to the AHB/ASB high speed peripherals and independently controllable. In order to reduce the number of clock names in a product, the HClocks are named MCK in the product datasheet.
- UHP Clock (UHPCK), required by USB Host Port operations.
- Programmable Clock Outputs can be selected from the clocks provided by the clock generator and driven on the PCKx pins.

25.2 Master Clock Controller

The Master Clock Controller provides selection and division of the Master Clock (MCK). MCK is the clock provided to all the peripherals and the memory controller.

The Master Clock is selected from one of the clocks provided by the Clock Generator. Selecting the Slow Clock provides a Slow Clock signal to the whole device. Selecting the Main Clock saves power consumption of the PLLs.

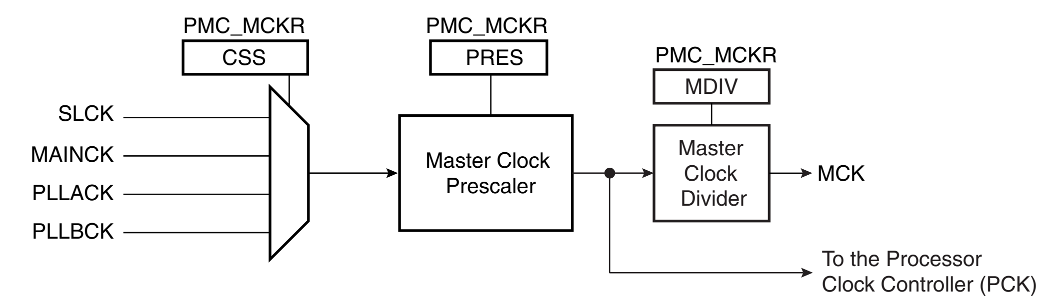
The Master Clock Controller is made up of a clock selector and a prescaler. It also contains a Master Clock divider which allows the processor clock to be faster than the Master Clock.

The Master Clock selection is made by writing the CSS field (Clock Source Selection) in PMC_MCKR (Master Clock Register). The prescaler supports the division by a power of 2 of the selected clock between 1 and 64. The PRES field in PMC_MCKR programs the prescaler. The Master Clock divider can be programmed through the MDIV field in PMC_MCKR.

Each time PMC_MCKR is written to define a new Master Clock, the MCKRDY bit is cleared in PMC_SR. It reads 0 until the Master Clock is established. Then, the MCKRDY bit is set and can trigger an interrupt to the processor. This feature is useful when switching from a high-speed clock to a lower one to inform the software when the change is actually done.



Figure 25-1. Master Clock Controller



25.3 Processor Clock Controller

The PMC features a Processor Clock Controller (PCK) that implements the Processor Idle Mode. The Processor Clock can be enabled and disabled by writing the System Clock Enable (PMC_SCER) and System Clock Disable Registers (PMC_SCDR). The status of this clock (at least for debug purpose) can be read in the System Clock Status Register (PMC_SCSR).

The Processor Clock PCK is enabled after a reset and is automatically re-enabled by any enabled interrupt. The Processor Idle Mode is achieved by disabling the Processor Clock, which is automatically re-enabled by any enabled fast or normal interrupt, or by the reset of the product.

When the Processor Clock is disabled, the current instruction is finished before the clock is stopped, but this does not prevent data transfers from other masters of the system bus.

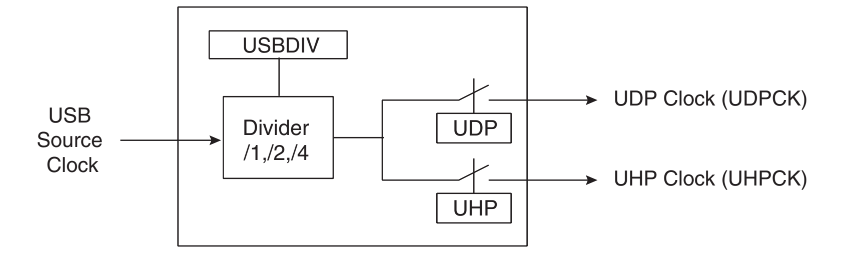
25.4 USB Clock Controller

The USB Source Clock is always generated from the PLL B output. If using the USB, the user must program the PLL to generate a 48 MHz, a 96 MHz or a 192 MHz signal with an accuracy of ± 0.25% depending on the USBDIV bit in CKGR_PLLBR (see Figure 25-6).

When the PLL B output is stable, i.e., the LOCKB is set:

- The USB host clock can be enabled by setting the UHP bit in PMC_SCER. To save power on this peripheral when it is not used, the user can set the UHP bit in PMC_SCDR. The UHP bit in PMC_SCSR gives the activity of this clock. The USB host port require both the 12/48 MHz signal and the Master Clock. The Master Clock may be controlled via the Peripheral Clock Controller.

Figure 25-2. USB Clock Controller



25.5 Peripheral Clock Controller

The Power Management Controller controls the clocks of each embedded peripheral by the way of the Peripheral Clock Controller. The user can individually enable and disable the Master Clock on the peripherals by writing into the Peripheral Clock Enable (PMC_PCER) and Peripheral Clock Disable (PMC_PCDR) registers. The status of the peripheral clock activity can be read in the Peripheral Clock Status Register (PMC_PCSR).

When a peripheral clock is disabled, the clock is immediately stopped. The peripheral clocks are automatically disabled after a reset.

In order to stop a peripheral, it is recommended that the system software wait until the peripheral has executed its last programmed operation before disabling the clock. This is to avoid data corruption or erroneous behavior of the system.

The bit number within the Peripheral Clock Control registers (PMC_PCER, PMC_PCDR, and PMC_PCSR) is the Peripheral Identifier defined at the product level. Generally, the bit number corresponds to the interrupt source number assigned to the peripheral.

25.6 HClock Controller

The PMC facilitates control of the clocks of each specific AHB/ASB peripheral by means of the HClock Controller. The user can individually enable and disable the Hclocks by writing into the registers; System Clock Enable (PMC_SCER) and System Clock Disable (PMC_SCDR). The status of HClock activity can be read in the System Clock Status Register (PMC_SCSR).

When an HClock is disabled, the clock is immediately stopped. When the HClock is re-enabled, the peripheral resumes action where it left off. The HClocks are automatically disabled after a reset.

1 HClock can be controlled.

25.7 Programmable Clock Output Controller

The PMC controls 4 signals to be output on external pins PCKx. Each signal can be independently programmed via the PMC_PCKx registers.

PCKx can be independently selected between the Slow clock, the PLL A output, the PLL B output and the main clock by writing the CSS field in PMC_PCKx. Each output signal can also be divided by a power of 2 between 1 and 64 by writing the PRES (Prescaler) field in PMC_PCKx.

Each output signal can be enabled and disabled by writing 1 in the corresponding bit, PCKx of PMC_SCER and PMC_SCDR, respectively. Status of the active programmable output clocks are given in the PCKx bits of PMC_SCSR (System Clock Status Register).

Moreover, like the PCK, a status bit in PMC_SR indicates that the Programmable Clock is actually what has been programmed in the Programmable Clock registers.

As the Programmable Clock Controller does not manage with glitch prevention when switching clocks, it is strongly recommended to disable the Programmable Clock before any configuration change and to re-enable it after the change is actually performed.

25.8 Programming Sequence

1. Enabling the Main Oscillator:

   The main oscillator is enabled by setting the MOSCEN field in the CKGR_MOR register. In some cases it may be advantageous to define a start-up time. This can be achieved by writing a value in the OSCOUNT field in the CKGR_MOR register.

   Once this register has been correctly configured, the user must wait for MOSCS field in the PMC_SR register to be set. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to MOSCS has been enabled in the PMC_IER register.

   Code Example:

   ```
   write_register(CKGR_MOR,0x00000701)
   ```

   Start Up Time = 8 * OSCOUNT / SLCK = 56 Slow Clock Cycles.

   So, the main oscillator will be enabled (MOSCS bit set) after 56 Slow Clock Cycles.

2. Checking the Main Oscillator Frequency (Optional):

   In some situations the user may need an accurate measure of the main oscillator frequency. This measure can be accomplished via the CKGR_MCFR register.

   Once the MAINRDY field is set in CKGR_MCFR register, the user may read the MAINF field in CKGR_MCFR register. This provides the number of main clock cycles within sixteen slow clock cycles.

3. Setting PLL A and divider A:

   All parameters necessary to configure PLL A and divider A are located in the CKGR_PLLAR register.

   It is important to note that Bit 29 must always be set to 1 when programming the CKGR_PLLAR register.

   The DIVA field is used to control the divider A itself. The user can program a value between 0 and 255. Divider A output is divider A input divided by DIVA. By default, DIVA parameter is set to 0 which means that divider A is turned off.

   The OUTA field is used to select the PLL A output frequency range.

   The MULA field is the PLL A multiplier factor. This parameter can be programmed between 0 and 2047. If MULA is set to 0, PLL A will be turned off. Otherwise PLL A output frequency is PLL A input frequency multiplied by (MULA + 1).

   The PLLACOUNT field specifies the number of slow clock cycles before LOCKA bit is set in the PMC_SR register after CKGR_PLLAR register has been written.

   Once CKGR_PLLAR register has been written, the user is obliged to wait for the LOCKA bit to be set in the PMC_SR register. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to LOCKA has been enabled in the PMC_IER register.

   All parameters in CKGR_PLLAR can be programmed in a single write operation. If at some stage one of the following parameters, SRCA, MULA, DIVA is modified, LOCKA bit will go

low to indicate that PLL A is not ready yet. When PLL A is locked, LOCKA will be set again. User has to wait for LOCKA bit to be set before using the PLL A output clock.

Code Example:

```
write_register(CKGR_PLLAR,0x20030605)
```

PLL A and divider A are enabled. PLL A input clock is main clock divided by 5. PLL An output clock is PLL A input clock multiplied by 4. Once CKGR_PLLAR has been written, LOCKA bit will be set after six slow clock cycles.

4. Setting PLL B and divider B:

All parameters needed to configure PLL B and divider B are located in the CKGR_PLLBR register.

The DIVB field is used to control divider B itself. A value between 0 and 255 can be programmed. Divider B output is divider B input divided by DIVB parameter. By default DIVB parameter is set to 0 which means that divider B is turned off.

The OUTB field is used to select the PLL B output frequency range.

The MULB field is the PLL B multiplier factor. This parameter can be programmed between 0 and 2047. If MULB is set to 0, PLL B will be turned off, otherwise the PLL B output frequency is PLL B input frequency multiplied by (MULB + 1).

The PLLBCOUNT field specifies the number of slow clock cycles before LOCKB bit is set in the PMC_SR register after CKGR_PLLBR register has been written.

Once the PMC_PLLB register has been written, the user must wait for the LOCKB bit to be set in the PMC_SR register. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to LOCKB has been enabled in the PMC_IER register. All parameters in CKGR_PLLBR can be programmed in a single write operation. If at some stage one of the following parameters, MULB, DIVB is modified, LOCKB bit will go low to indicate that PLL B is not ready yet. When PLL B is locked, LOCKB will be set again. The user is constrained to wait for LOCKB bit to be set before using the PLL A output clock.

The USBDIV field is used to control the additional divider by 1, 2 or 4, which generates the USB clock(s).

Code Example:

```
write_register(CKGR_PLLBR,0x00040805)
```

If PLL B and divider B are enabled, the PLL B input clock is the main clock. PLL B output clock is PLL B input clock multiplied by 5. Once CKGR_PLLBR has been written, LOCKB bit will be set after eight slow clock cycles.

5. Selection of Master Clock and Processor Clock

The Master Clock and the Processor Clock are configurable via the PMC_MCKR register.

The CSS field is used to select the Master Clock divider source. By default, the selected clock source is slow clock.

The PRES field is used to control the Master Clock prescaler. The user can choose between different values (1, 2, 4, 8, 16, 32, 64). Master Clock output is prescaler input



divided by PRES parameter. By default, PRES parameter is set to 1 which means that master clock is equal to slow clock.

The MDIV field is used to control the Master Clock prescaler. It is possible to choose between different values (0, 1, 2). The Master Clock output is Processor Clock divided by 1, 2 or 4, depending on the value programmed in MDIV. By default, MDIV is set to 0, which indicates that the Processor Clock is equal to the Master Clock.

Once the PMC_MCKR register has been written, the user must wait for the MCKRDY bit to be set in the PMC_SR register. This can be done either by polling the status register or by waiting for the interrupt line to be raised if the associated interrupt to MCKRDY has been enabled in the PMC_IER register.

The PMC_MCKR register must not be programmed in a single write operation. The preferred programming sequence for the PMC_MCKR register is as follows:

- If a new value for CSS field corresponds to PLL Clock,
  - Program the PRES field in the PMC_MCKR register.
  - Wait for the MCKRDY bit to be set in the PMC_SR register.
  - Program the CSS field in the PMC_MCKR register.
  - Wait for the MCKRDY bit to be set in the PMC_SR register.
- If a new value for CSS field corresponds to Main Clock or Slow Clock,
  - Program the CSS field in the PMC_MCKR register.
  - Wait for the MCKRDY bit to be set in the PMC_SR register.
  - Program the PRES field in the PMC_MCKR register.
  - Wait for the MCKRDY bit to be set in the PMC_SR register.

If at some stage one of the following parameters, CSS or PRES, is modified, the MCKRDY bit will go low to indicate that the Master Clock and the Processor Clock are not ready yet. The user must wait for MCKRDY bit to be set again before using the Master and Processor Clocks.

Note: IF PLLx clock was selected as the Master Clock and the user decides to modify it by writing in CKGR_PLLR (CKGR_PLLAR or CKGR_PLLBR), the MCKRDY flag will go low while PLL is unlocked. Once PLL is locked again, LOCK (LOCKA or LOCKB) goes high and MCKRDY is set. While PLLA is unlocked, the Master Clock selection is automatically changed to Slow Clock.. While PLLB is unlocked, the Master Clock selection is automatically changed to Main Clock. For further information, see Section 25.10.2. ”Clock Switching Waveforms” on page 238.

Code Example:

```
write_register(PMC_MCKR,0x00000001)
wait (MCKRDY=1)
write_register(PMC_MCKR,0x00000011)
wait (MCKRDY=1)
```

The Master Clock is main clock divided by 16.

The Processor Clock is the Master Clock.



6. Selection of Programmable clocks

Programmable clocks are controlled via registers; PMC_SCER, PMC_SCDR and PMC_SCSR.

Programmable clocks can be enabled and/or disabled via the PMC_SCER and PMC_SCDR registers. Depending on the system used, 4 Programmable clocks can be enabled or disabled. The PMC_SCSR provides a clear indication as to which Programmable clock is enabled. By default all Programmable clocks are disabled.

PMC_PCKx registers are used to configure Programmable clocks.

The CSS field is used to select the Programmable clock divider source. Four clock options are available: main clock, slow clock, PLLACK, PLLBCK. By default, the clock source selected is slow clock.

The PRES field is used to control the Programmable clock prescaler. It is possible to choose between different values (1, 2, 4, 8, 16, 32, 64). Programmable clock output is prescaler input divided by PRES parameter. By default, the PRES parameter is set to 1 which means that master clock is equal to slow clock.

Once the PMC_PCKx register has been programmed, The corresponding Programmable clock must be enabled and the user is constrained to wait for the PCKRDYx bit to be set in the PMC_SR register. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to PCKRDYx has been enabled in the PMC_IER register. All parameters in PMC_PCKx can be programmed in a single write operation.

If the CSS and PRES parameters are to be modified, the corresponding Programmable clock must be disabled first. The parameters can then be modified. Once this has been done, the user must re-enable the Programmable clock and wait for the PCKRDYx bit to be set.

Code Example:

```
write_register(PMC_PCK0,0x00000015)
```

Programmable clock 0 is main clock divided by 32.

7. Enabling Peripheral Clocks

Once all of the previous steps have been completed, the peripheral clocks can be enabled and/or disabled via registers PMC_PCER and PMC_PCDR.

Depending on the system used, 17 peripheral clocks can be enabled or disabled. The PMC_PCSR provides a clear view as to which peripheral clock is enabled.

Note: Each enabled peripheral clock corresponds to Master Clock.

Code Examples:

```
write_register(PMC_PCER,0x00000110)
```

Peripheral clocks 4 and 8 are enabled.

```
write_register(PMC_PCDR,0x00000010)
```

Peripheral clock 4 is disabled.

8. Enabling HClocks

   Once all of the previous steps have been completed, the HClocks can be enabled and/or disabled via registers; PMC_SCER and PMC_SCDR.

   Depending on the system used, 1 HClocks can be enabled or disabled.

   The PMC_SCSR register indicates which HClock is enabled.

Note: Each enabled HClock corresponds to Master Clock.

Code Examples:

```
write_register(PMC_SCER,0x00110000)
```

HClocks 0 and 4 are enabled.

25.9 Clock Switching Details

25.9.1 Master Clock Switching Timings

Table 25-1 and Table 25-2 give the worst case timings required for the Master Clock to switch from one selected clock to another one. This is in the event that the prescaler is de-activated. When the prescaler is activated, an additional time of 64 clock cycles of the new selected clock has to be added.

Table 25-1. Clock Switching Timings (Worst Case)

From To	Main Clock	SLCK	PLL Clock
Main Clock	–	4 x SLCK + 2.5 x Main Clock	3 x PLL Clock + 4 x SLCK + 1 x Main Clock
SLCK	0.5 x Main Clock + 4.5 x SLCK	–	3 x PLL Clock + 5 x SLCK
PLL Clock	0.5 x Main Clock + 4 x SLCK + PLLCOUNT x SLCK + 2.5 x PLLx Clock	2.5 x PLL Clock + 5 x SLCK + PLLCOUNT x SLCK	2.5 x PLL Clock + 4 x SLCK + PLLCOUNT x SLCK

Notes: 1. PLL designates either the PLL A or the PLL B Clock.
2. PLLCOUNT designates either PLLACOUNT or PLLBCOUNT.

Table 25-2. Clock Switching Timings Between Two PLLs (Worst Case)

From To	PLLA Clock	PLLB Clock
PLLA Clock	2.5 x PLLA Clock + 4 x SLCK + PLLACOUNT x SLCK	3 x PLLA Clock + 4 x SLCK + 1.5 x PLLA Clock
PLLB Clock	3 x PLLB Clock + 4 x SLCK + 1.5 x PLLB Clock	2.5 x PLLB Clock + 4 x SLCK + PLLBCOUNT x SLCK

25.9.2 Clock Switching Waveforms

Figure 25-3. Switch Master Clock from Slow Clock to PLL Clock

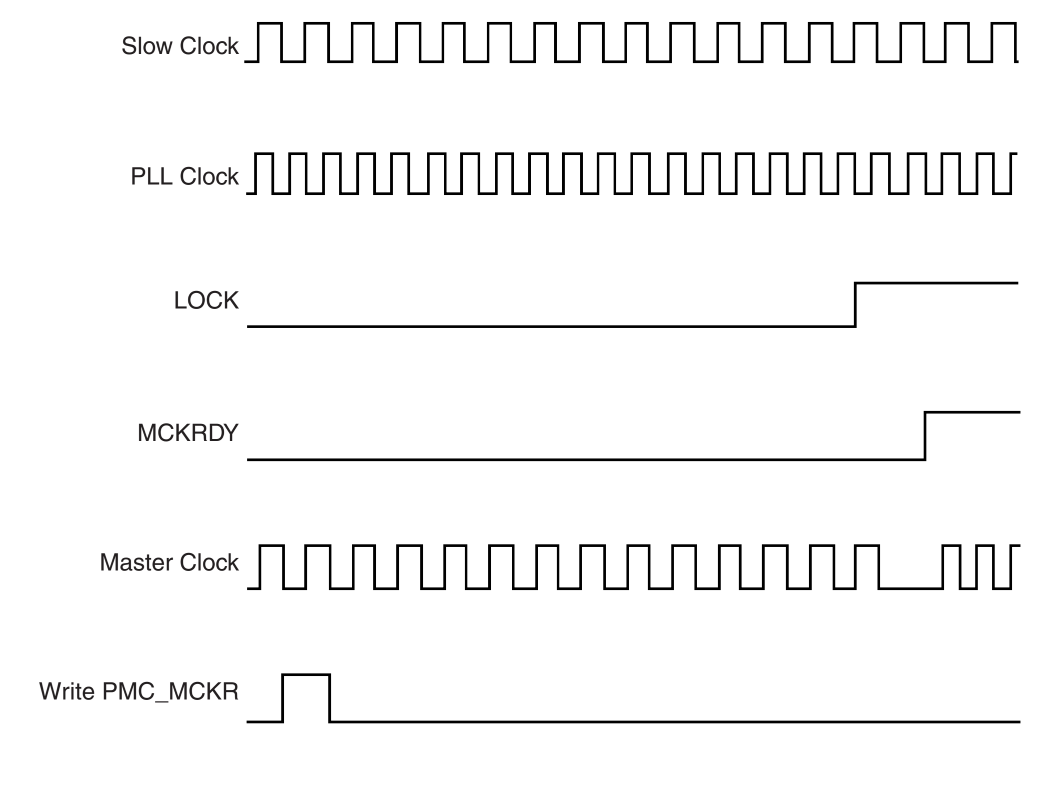


Figure 25-4. Switch Master Clock from Main Clock to Slow Clock

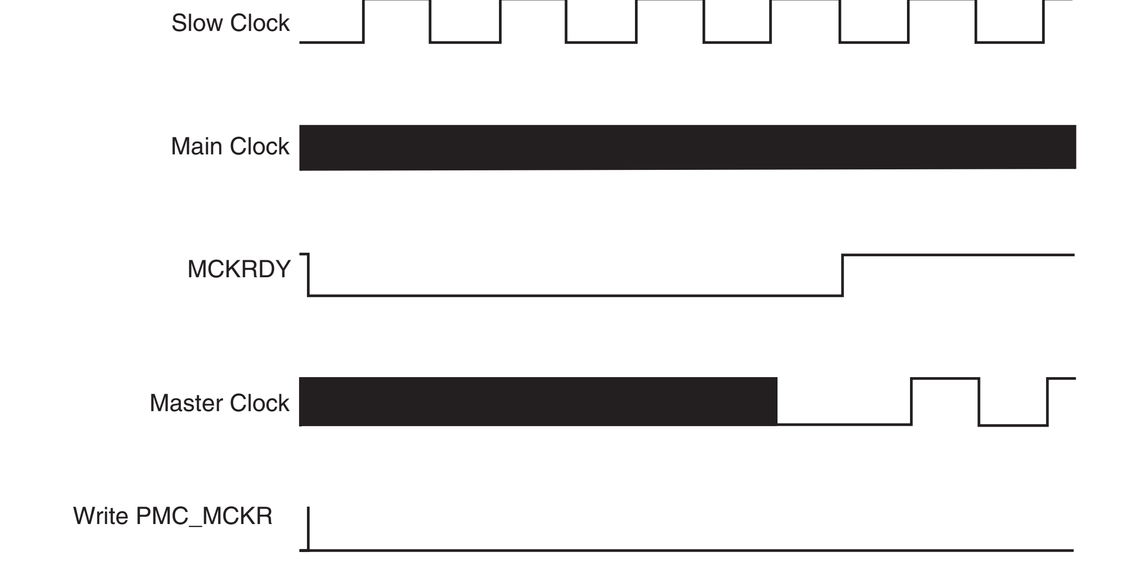


Figure 25-5. Change PLLA Programming

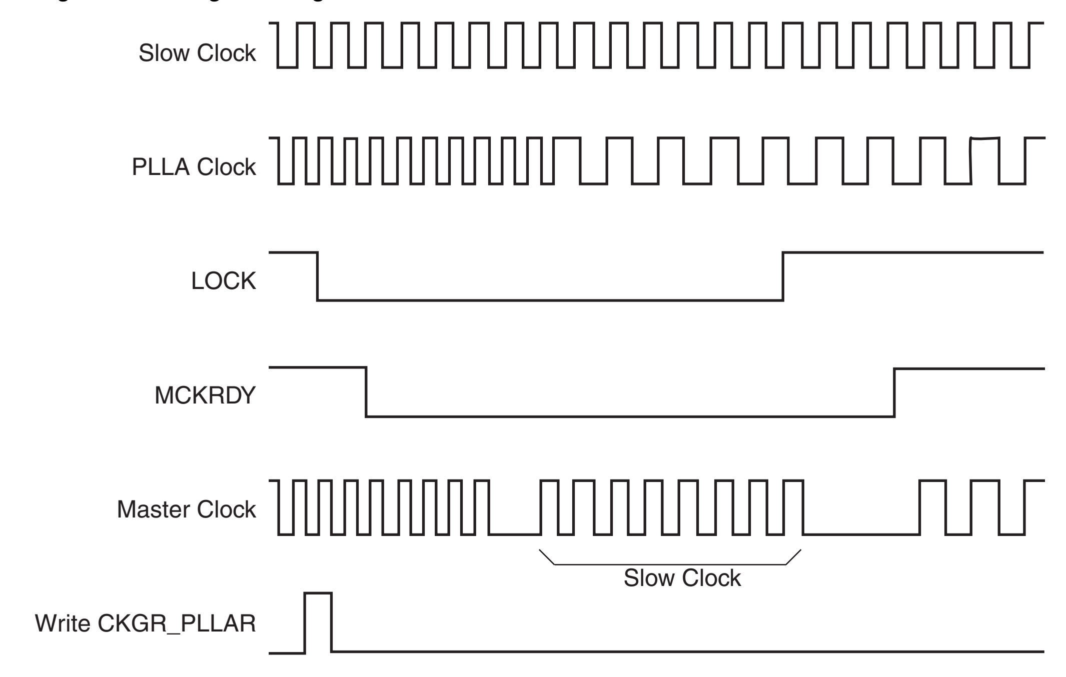


Figure 25-6. Change PLLB Programming

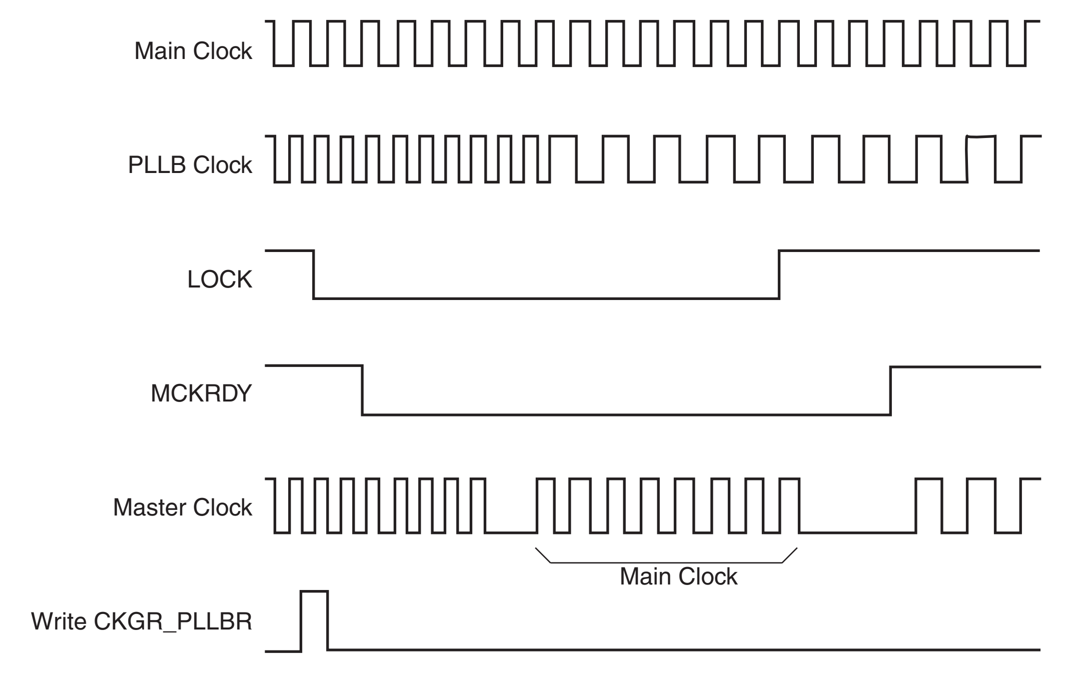
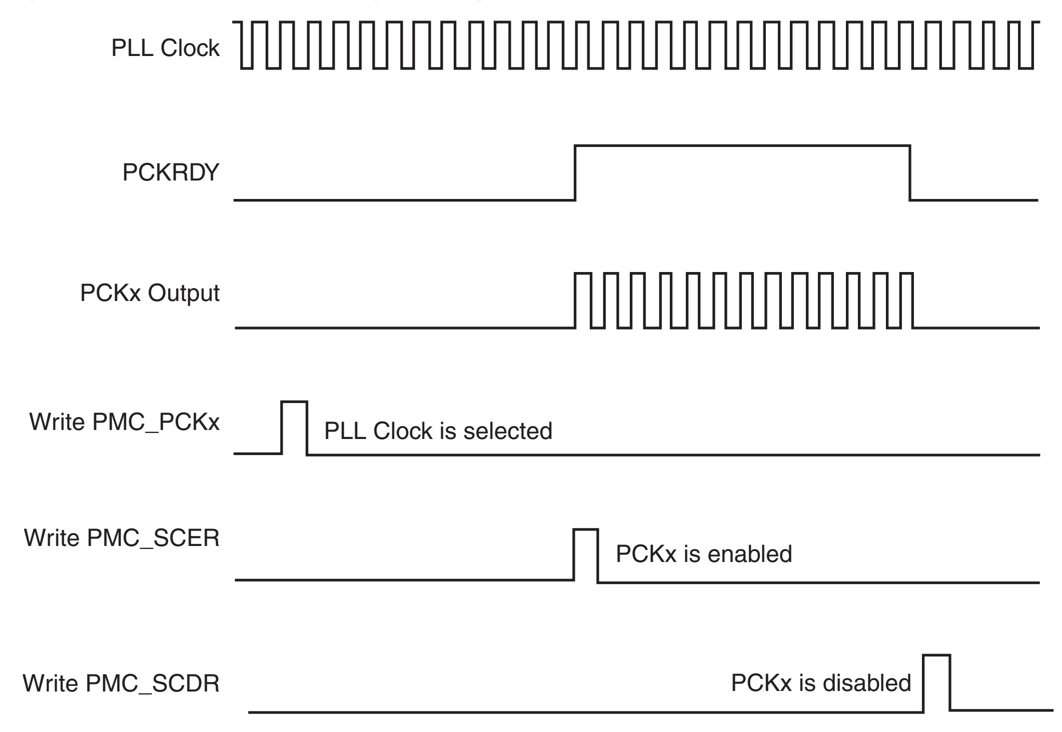


Figure 25-7. Programmable Clock Output Programming



25.10 Power Management Controller (PMC) User Interface

Table 25-3. Register Mapping

Offset	Register	Name	Access	Reset Value
0x0000	System Clock Enable Register	PMC_SCER	Write-only	–
0x0004	System Clock Disable Register	PMC_SCDR	Write-only	–
0x0008	System Clock Status Register	PMC _SCSR	Read-only	0x03
0x000C	Reserved	–	–	–
0x0010	Peripheral Clock Enable Register	PMC _PCER	Write-only	–
0x0014	Peripheral Clock Disable Register	PMC_PCDR	Write-only	–
0x0018	Peripheral Clock Status Register	PMC_PCSR	Read-only	0x0
0x001C	Reserved	–	–	–
0x0020	Main Oscillator Register	CKGR_MOR	Read/Write	0x0
0x0024	Main Clock Frequency Register	CKGR_MCFR	Read-only	0x0
0x0028	PLL A Register	CKGR_PLLAR	ReadWrite	0x3F00
0x002C	PLL B Register	CKGR_PLLBR	ReadWrite	0x3F00
0x0030	Master Clock Register	PMC_MCKR	Read/Write	0x0
0x0034	Application Clock Register	PMC_ACKR	Read/Write	0x0
0x0038	Reserved	–	–	–
0x003C	Reserved	–	–	–
0x0040	Programmable Clock 0 Register	PMC_PCK0	Read/Write	0x0
0x0044	Programmable Clock 1 Register	PMC_PCK1	Read/Write	0x0
...	...	...	...	...
0x0060	Interrupt Enable Register	PMC_IER	Write-only	--
0x0064	Interrupt Disable Register	PMC_IDR	Write-only	--
0x006C	Interrupt Mask Register	PMC_IMR	Read-only	0x0
0x0070 - 0x00FC	Reserved	–	–	–

25.10.1 PMC System Clock Enable Register

Register Name: PMC_SCER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	HCK0

15	14	13	12	11	10	9	8
–	–	–	–	PCK3	PCK2	PCK1	PCK0

7	6	5	4	3	2	1	0
UDP	UHP	–	–	–	–	–	PCK

- **PCK: Processor Clock Enable**

0 = No effect.

1 = Enables the Processor clock.

- **UHP: USB Host Port Clock Enable**

0 = No effect.

1 = Enables the 12 and 48 MHz clock of the USB Host Port.

- **UDP: USB Device Port Clock Enable**

0 = No effect.

1 = Enables the 48 MHz clock of the USB Device Port.

- **PCKx: Programmable Clock x Output Enable**

0 = No effect.

1 = Enables the corresponding Programmable Clock output.

- **HCKx: HClock x Output Enable**

0 = No effect.

1 = Enables the corresponding HClock output.



25.10.2 PMC System Clock Disable Register

Register Name: PMC_SCDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	HCK0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	PCK3	PCK2	PCK1	PCK0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UDP	UHP	–	–	–	–	–	PCK

- **PCK: Processor Clock Disable**

0 = No effect.

1 = Disables the Processor clock. This is used to enter the processor in Idle Mode.

- **UHP: USB Host Port Clock Disable**

0 = No effect.

1 = Disables the 12 and 48 MHz clock of the USB Host Port.

- **UDP: USB Device Port Clock Disable**

0 = No effect.

1 = Disables the 48 MHz clock of the USB Device Port.

- **PCKx: Programmable Clock x Output Disable**

0 = No effect.

1 = Disables the corresponding Programmable Clock output.

- **HCKx: Hclock x Output Disable**

0 = No effect.

1 = Disables the corresponding HClock output.



25.10.3 PMC System Clock Status Register

Register Name: PMC_SCSR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	HCK0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	PCK3	PCK2	PCK1	PCK0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UDP	UHP	–	–	–	–	–	PCK

- **PCK: Processor Clock Status**

0 = The Processor clock is disabled.

1 = The Processor clock is enabled.

- **UHP: USB Host Port Clock Status**

0 = The 12 and 48 MHz clock (UHPCK) of the USB Host Port is disabled.

1 = The 12 and 48 MHz clock (UHPCK) of the USB Host Port is enabled.

- **UDP: USB Device Port Clock Status**

0 = The 48 MHz clock (UDPCK) of the USB Device Port is disabled.

1 = The 48 MHz clock (UDPCK) of the USB Device Port is enabled.

- **PCKx: Programmable Clock x Output Status**

0 = The corresponding Programmable Clock output is disabled.

1 = The corresponding Programmable Clock output is enabled.

- **HCKx: HClock Output x Status**

0 = The corresponding HClock output is disabled.

1 = The corresponding HClock output is enabled.

25.10.4 PMC Peripheral Clock Enable Register

Register Name: PMC_PCER

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	-	-

- **PIDx: Peripheral Clock x Enable**

0 = No effect.

1 = Enables the corresponding peripheral clock.

Note: Programming the control bits of the Peripheral ID that are not implemented has no effect on the behavior of the PMC.

25.10.5 PMC Peripheral Clock Disable Register

Register Name: PMC_PCDR

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	-	-

- **PIDx: Peripheral Clock x Disable**

0 = No effect.

1 = Disables the corresponding peripheral clock.



25.10.6 PMC Peripheral Clock Status Register

Register Name: PMC_PCSR

Access Type: Read-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PID7	PID6	PID5	PID4	PID3	PID2	–	–

- **PIDx: Peripheral Clock x Status**

0 = The corresponding peripheral clock is disabled.

1 = The corresponding peripheral clock is enabled.

25.10.7 PMC Clock Generator Main Oscillator Register

Register Name: CKGR_MOR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
OSCOUNT							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	OSCBYPASS	MOSCEN

- **MOSCEN: Main Oscillator Enable**

A crystal must be connected between XIN and XOUT.

0 = The Main Oscillator is disabled.

1 = The Main Oscillator is enabled. OSCBYPASS must be set to 0.

When MOSCEN is set, the MOSCS flag is set once the Main Oscillator startup time is achieved.

- **OSCBYPASS: Oscillator Bypass**

0 = No effect.

1 = The Main Oscillator is bypassed. MOSCEN must be set to 0. An external clock must be connected on XIN.

When OSCBYPASS is set, the MOSCS flag in PMC_SR is automatically set.

Clearing MOSCEN and OSCBYPASS bits allows resetting the MOSCS flag.

- **OSCOUNT: Main Oscillator Start-up Time**

Specifies the number of Slow Clock cycles multiplied by 8 for the Main Oscillator start-up time.

25.10.8 PMC Clock Generator Main Clock Frequency Register

Register Name: CKGR_MCFR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	MAINRDY

15	14	13	12	11	10	9	8
MAINF							

7	6	5	4	3	2	1	0
MAINF							

- **MAINF: Main Clock Frequency**

Gives the number of Main Clock cycles within 16 Slow Clock periods.

- **MAINRDY: Main Clock Ready**

0 = MAINF value is not valid or the Main Oscillator is disabled.

1 = The Main Oscillator has been enabled previously and MAINF value is available.

25.10.9 PMC Clock Generator PLL A Register

Register Name: CKGR_PLLAR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	1	–	–	MULA		

23	22	21	20	19	18	17	16
MULA							

15	14	13	12	11	10	9	8
OUTA		PLLACOUNT					

7	6	5	4	3	2	1	0
DIVA							

Possible limitations on PLL A input frequencies and multiplier factors should be checked before using the PMC.

Warning: Bit 29 must always be set to 1 when programming the CKGR_PLLAR register.

- **DIVA: Divider A**

DIVA	Divider Selected
0	Divider output is 0
1	Divider is bypassed
2 - 255	Divider output is the Main Clock divided by DIVA.

- **PLLACOUNT: PLL A Counter**

Specifies the number of Slow Clock cycles before the LOCKA bit is set in PMC_SR after CKGR_PLLAR is written.

- **OUTA: PLL A Clock Frequency Range**

OUTA		PLL A Frequency Output Range
0	0	Refer to the DC Characteristics section of the product datasheet
0	1	Reserved
1	0	Refer to the DC Characteristics section of the product datasheet
1	1	Reserved

- **MULA: PLL A Multiplier**

0 = The PLL A is deactivated.

1 up to 2047 = The PLL A Clock frequency is the PLL A input frequency multiplied by MULA + 1.

25.10.10 PMC Clock Generator PLL B Register

Register Name: CKGR_PLLBR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	USBDIV		–	MULB		

23	22	21	20	19	18	17	16
MULB							

15	14	13	12	11	10	9	8
OUTB		PLLBCOUNT					

7	6	5	4	3	2	1	0
DIVB							

Possible limitations on PLL B input frequencies and multiplier factors should be checked before using the PMC.

- **DIVB: Divider B**

DIVB	Divider Selected
0	Divider output is 0
1	Divider is bypassed
2 - 255	Divider output is the selected clock divided by DIVB.

- **PLLBCOUNT: PLL B Counter**

Specifies the number of slow clock cycles before the LOCKB bit is set in PMC_SR after CKGR_PLLBR is written.

- **OUTB: PLLB Clock Frequency Range**

OUTB		PLL B Clock Frequency Range
0	0	Refer to the DC Characteristics section of the product datasheet
0	1	Reserved
1	0	Refer to the DC Characteristics section of the product datasheet
1	1	Reserved

- **MULB: PLL Multiplier**

0 = The PLL B is deactivated.

1 up to 2047 = The PLL B Clock frequency is the PLL B input frequency multiplied by MULB + 1.

- **USBDIV: Divider for USB Clock**

USBDIV		Divider for USB Clock(s)
0	0	Divider output is PLL B clock output.
0	1	Divider output is PLL B clock output divided by 2.
1	0	Divider output is PLL B clock output divided by 4.
1	1	Reserved.



25.10.11 PMC Master Clock Register

Register Name: PMC_MCKR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	MDIV	

7	6	5	4	3	2	1	0
–	–	–	PRES			CSS	

- **CSS: Master Clock Selection**

CSS		Clock Source Selection
0	0	Slow Clock is selected
0	1	Main Clock is selected
1	0	PLL A Clock is selected
1	1	PLL B Clock is selected

- **PRES: Master Clock Prescaler**

PRES			Master Clock
0	0	0	Selected clock
0	0	1	Selected clock divided by 2
0	1	0	Selected clock divided by 4
0	1	1	Selected clock divided by 8
1	0	0	Selected clock divided by 16
1	0	1	Selected clock divided by 32
1	1	0	Selected clock divided by 64
1	1	1	Reserved

- **MDIV: Master Clock Division**

MDIV		Master Clock Division
0	0	Master Clock is Processor Clock.
0	1	Master Clock is Processor Clock divided by 2.
1	0	Master Clock is Processor Clock divided by 4.
1	1	Reserved.



25.10.12 PMC Programmable Clock Register

Register Name: PMC_PCKx

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	PRES			CSS	

- **CSS: Master Clock Selection**

CSS		Clock Source Selection
0	0	Slow Clock is selected
0	1	Main Clock is selected
1	0	PLL A Clock is selected
1	1	PLL B Clock is selected

- **PRES: Programmable Clock Prescaler**

PRES			Master Clock
0	0	0	Selected clock
0	0	1	Selected clock divided by 2
0	1	0	Selected clock divided by 4
0	1	1	Selected clock divided by 8
1	0	0	Selected clock divided by 16
1	0	1	Selected clock divided by 32
1	1	0	Selected clock divided by 64
1	1	1	Reserved

25.10.13 PMC Interrupt Enable Register

Register Name: PMC_IER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	PCKRDY3	PCKRDY2	PCKRDY1	PCKRDY0

7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Enable**
- **LOCKA: PLL A Lock Interrupt Enable**
- **LOCKB: PLL B Lock Interrupt Enable**
- **MCKRDY: Master Clock Ready Interrupt Enable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

25.10.14 PMC Interrupt Disable Register

Register Name: PMC_IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	PCKRDY3	PCKRDY2	PCKRDY1	PCKRDY0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Disable**
- **LOCKA: PLL A Lock Interrupt Disable**
- **LOCKB: PLL B Lock Interrupt Disable**
- **MCKRDY: Master Clock Ready Interrupt Disable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.



25.10.15 PMC Status Register

Register Name: PMC_SR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	PCKRDY3	PCKRDY2	PCKRDY1	PCKRDY0

7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: MOSCS Flag Status**

0 = Main oscillator is not stabilized.

1 = Main oscillator is stabilized.

- **LOCKA: PLL A Lock Status**

0 = PLL A is not locked

1 = PLL A is locked.

- **LOCKB: PLL B Lock Status**

0 = PLL B is not locked.

1 = PLL B is locked.

- **MCKRDY: Master Clock Status**

0 = Master Clock is not ready.

1 = Master Clock is ready.

- **PCKRDYx: Programmable Clock Ready Status**

0 = Programmable Clock x is not ready.

1 = Programmable Clock x is ready.



25.10.16 PMC Interrupt Mask Register

Register Name: PMC_IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	PCKRDY3	PCKRDY2	PCKRDY1	PCKRDY0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Mask**
- **LOCKA: PLL A Lock Interrupt Mask**
- **LOCKB: PLL B Lock Interrupt Mask**
- **MCKRDY: Master Clock Ready Interrupt Mask**
- **PCKRDYx: Programmable Clock Ready x Interrupt Mask**

0 = The corresponding interrupt is enabled.

1 = The corresponding interrupt is disabled.

26. Advanced Interrupt Controller (AIC)

26.1 Description

The Advanced Interrupt Controller (AIC) is an 8-level priority, individually maskable, vectored interrupt controller, providing handling of up to thirty-two interrupt sources. It is designed to substantially reduce the software and real-time overhead in handling internal and external interrupts.

The AIC drives the nFIQ (fast interrupt request) and the nIRQ (standard interrupt request) inputs of an ARM processor. Inputs of the AIC are either internal peripheral interrupts or external interrupts coming from the product's pins.

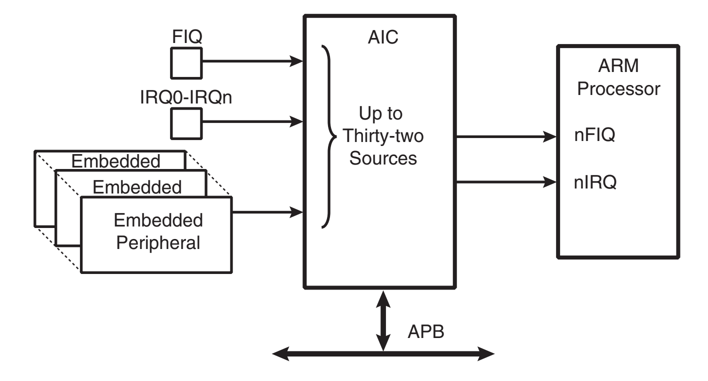
The 8-level Priority Controller allows the user to define the priority for each interrupt source, thus permitting higher priority interrupts to be serviced even if a lower priority interrupt is being treated.

Internal interrupt sources can be programmed to be level sensitive or edge triggered. External interrupt sources can be programmed to be positive-edge or negative-edge triggered or high-level or low-level sensitive.

The fast forcing feature redirects any internal or external interrupt source to provide a fast interrupt rather than a normal interrupt.

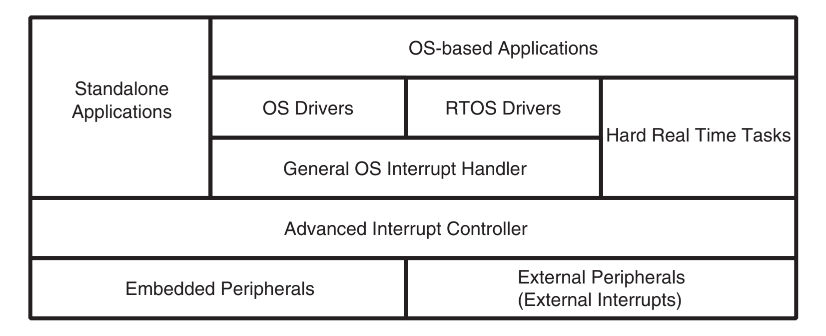
26.2 Block Diagram

Figure 26-1. Block Diagram



26.3 Application Block Diagram

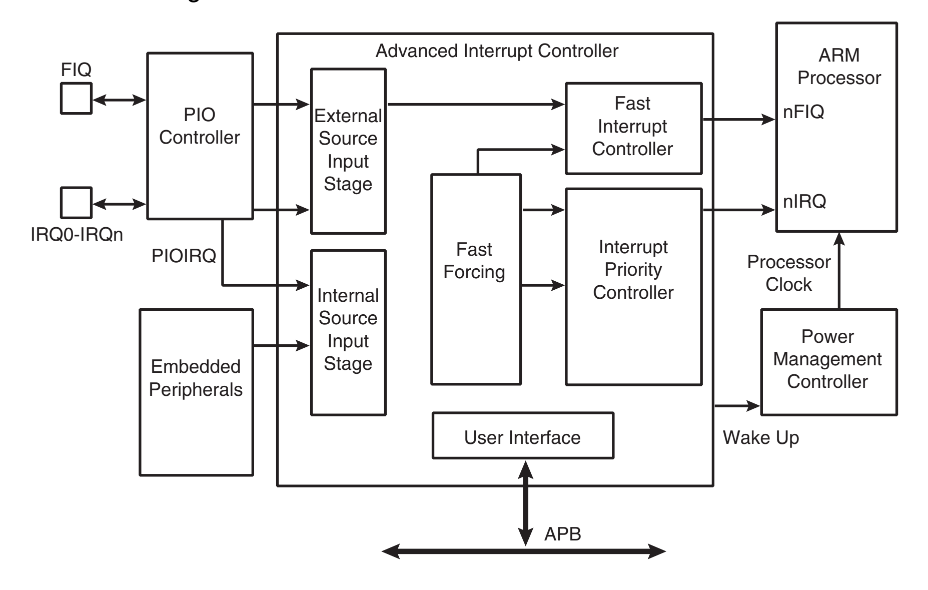
Figure 26-2. Description of the Application Block





26.4 AIC Detailed Block Diagram

Figure 26-3. AIC Detailed Block Diagram



26.5 I/O Line Description

Table 26-1. I/O Line Description

Pin Name	Pin Description	Type
FIQ	Fast Interrupt	Input
IRQ0 - IRQn	Interrupt 0 - Interrupt n	Input

26.6 Product Dependencies

26.6.1 I/O Lines

The interrupt signals FIQ and IRQ0 to IRQn are normally multiplexed through the PIO controllers. Depending on the features of the PIO controller used in the product, the pins must be programmed in accordance with their assigned interrupt function. This is not applicable when the PIO controller used in the product is transparent on the input path.

26.6.2 Power Management

The Advanced Interrupt Controller is continuously clocked. The Power Management Controller has no effect on the Advanced Interrupt Controller behavior.

The assertion of the Advanced Interrupt Controller outputs, either nIRQ or nFIQ, wakes up the ARM processor while it is in Idle Mode. The General Interrupt Mask feature enables the AIC to wake up the processor without asserting the interrupt line of the processor, thus providing synchronization of the processor on an event.

26.6.3 Interrupt Sources

The Interrupt Source 0 is always located at FIQ. If the product does not feature an FIQ pin, the Interrupt Source 0 cannot be used.

The Interrupt Source 1 is always located at System Interrupt. This is the result of the OR-wiring of the system peripheral interrupt lines, such as the System Timer, the Real Time Clock, the Power Management Controller and the Memory Controller. When a system interrupt occurs, the service routine must first distinguish the cause of the interrupt. This is performed by reading successively the status registers of the above mentioned system peripherals.

The interrupt sources 2 to 31 can either be connected to the interrupt outputs of an embedded user peripheral or to external interrupt lines. The external interrupt lines can be connected directly, or through the PIO Controller.

The PIO Controllers are considered as user peripherals in the scope of interrupt handling. Accordingly, the PIO Controller interrupt lines are connected to the Interrupt Sources 2 to 31.

The peripheral identification defined at the product level corresponds to the interrupt source number (as well as the bit number controlling the clock of the peripheral). Consequently, to simplify the description of the functional operations and the user interface, the interrupt sources are named FIQ, SYS, and PID2 to PID31.

26.7 Functional Description

26.7.1 Interrupt Source Control

26.7.1.1 Interrupt Source Mode

The Advanced Interrupt Controller independently programs each interrupt source. The SRCTYPE field of the corresponding AIC_SMR (Source Mode Register) selects the interrupt condition of each source.

The internal interrupt sources wired on the interrupt outputs of the embedded peripherals can be programmed either in level-sensitive mode or in edge-triggered mode. The active level of the internal interrupts is not important for the user.

The external interrupt sources can be programmed either in high level-sensitive or low level-sensitive modes, or in positive edge-triggered or negative edge-triggered modes.

26.7.1.2 Interrupt Source Enabling

Each interrupt source, including the FIQ in source 0, can be enabled or disabled by using the command registers; AIC_IECR (Interrupt Enable Command Register) and AIC_IDCR (Interrupt Disable Command Register). This set of registers conducts enabling or disabling in one instruction. The interrupt mask can be read in the AIC_IMR register. A disabled interrupt does not affect servicing of other interrupts.

26.7.1.3 Interrupt Clearing and Setting

All interrupt sources programmed to be edge-triggered (including the FIQ in source 0) can be individually set or cleared by writing respectively the AIC_ISCR and AIC_ICCR registers. Clearing or setting interrupt sources programmed in level-sensitive mode has no effect.

The clear operation is perfunctory, as the software must perform an action to reinitialize the "memorization" circuitry activated when the source is programmed in edge-triggered mode. However, the set operation is available for auto-test or software debug purposes. It can also be used to execute an AIC-implementation of a software interrupt.

The AIC features an automatic clear of the current interrupt when the AIC_IVR (Interrupt Vector Register) is read. Only the interrupt source being detected by the AIC as the current

interrupt is affected by this operation. (See "Priority Controller" on page 265.) The automatic clear reduces the operations required by the interrupt service routine entry code to reading the AIC_IVR. Note that the automatic interrupt clear is disabled if the interrupt source has the Fast Forcing feature enabled as it is considered uniquely as a FIQ source. (For further details, See "Fast Forcing" on page 269.)

The automatic clear of the interrupt source 0 is performed when AIC_FVR is read.

26.7.1.4 *Interrupt Status*

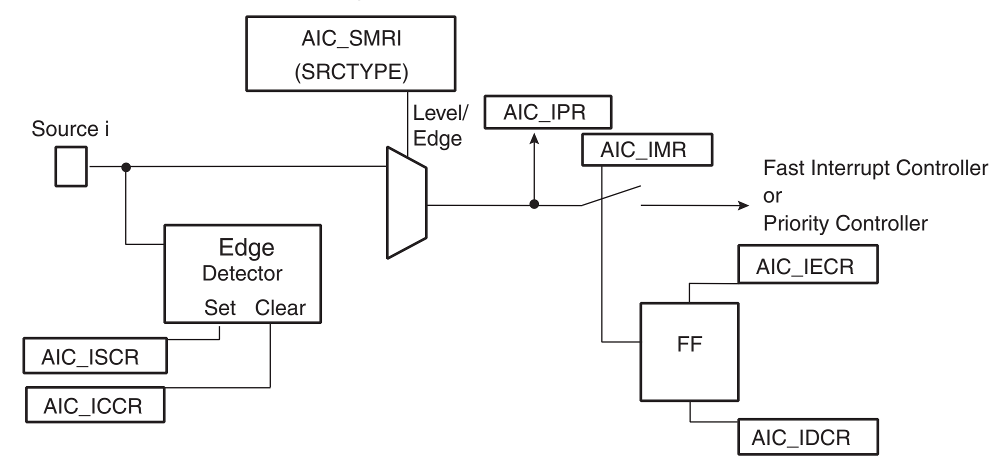
For each interrupt, the AIC operation originates in AIC_IPR (Interrupt Pending Register) and its mask in AIC_IMR (Interrupt Mask Register). AIC_IPR enables the actual activity of the sources, whether masked or not.

The AIC_ISR register reads the number of the current interrupt (see "Priority Controller" on page 265) and the register AIC_CISR gives an image of the signals nIRQ and nFIQ driven on the processor.

Each status referred to above can be used to optimize the interrupt handling of the systems.

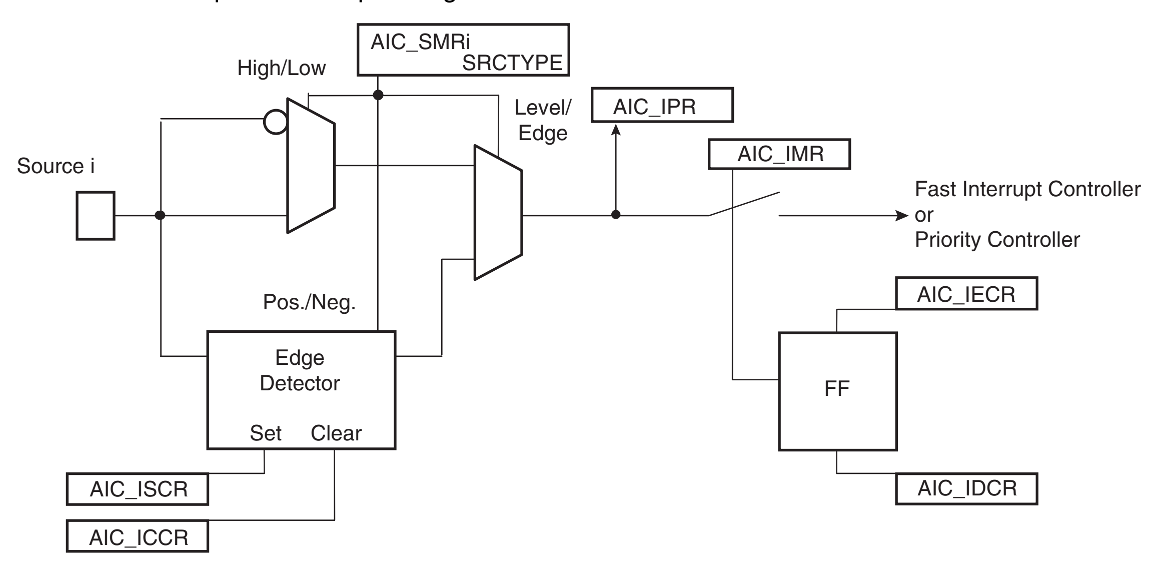
26.7.1.5 *Internal Interrupt Source Input Stage*

Figure 26-4. Internal Interrupt Source Input Stage



26.7.1.6 External Interrupt Source Input Stage

Figure 26-5. External Interrupt Source Input Stage



26.7.2 Interrupt Latencies

Global interrupt latencies depend on several parameters, including:

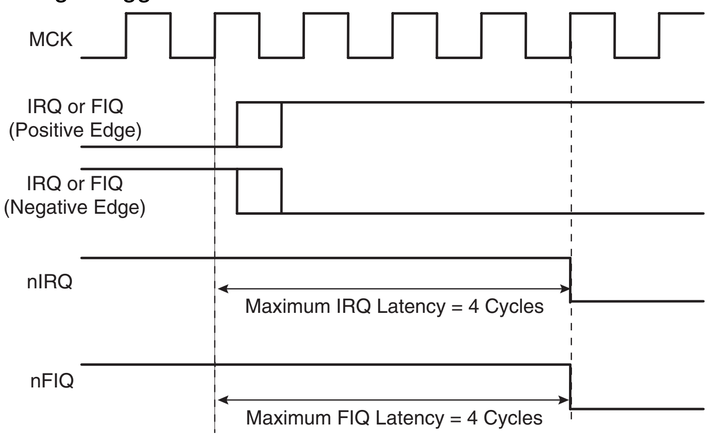
- The time the software masks the interrupts.
- Occurrence, either at the processor level or at the AIC level.
- The execution time of the instruction in progress when the interrupt occurs.
- The treatment of higher priority interrupts and the resynchronization of the hardware signals.

This section addresses only the hardware resynchronizations. It gives details of the latency times between the event on an external interrupt leading in a valid interrupt (edge or level) or the assertion of an internal interrupt source and the assertion of the nIRQ or nFIQ line on the processor. The resynchronization time depends on the programming of the interrupt source and on its type (internal or external). For the standard interrupt, resynchronization times are given assuming there is no higher priority in progress.

The PIO Controller multiplexing has no effect on the interrupt latencies of the external interrupt sources.

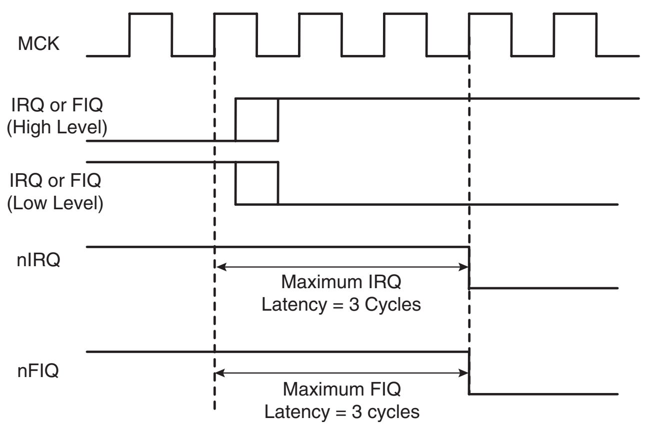
26.7.2.1 External Interrupt Edge Triggered Source

Figure 26-6. External Interrupt Edge Triggered Source



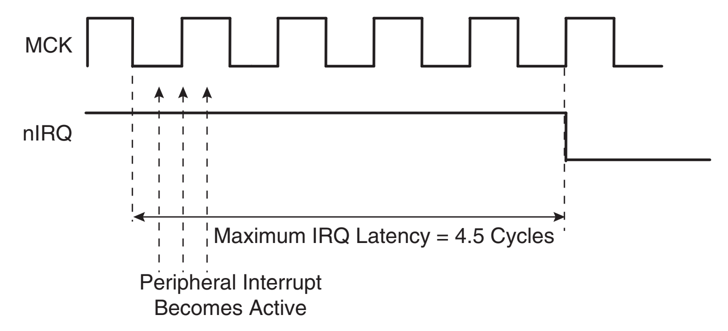
26.7.2.2 External Interrupt Level Sensitive Source

Figure 26-7. External Interrupt Level Sensitive Source



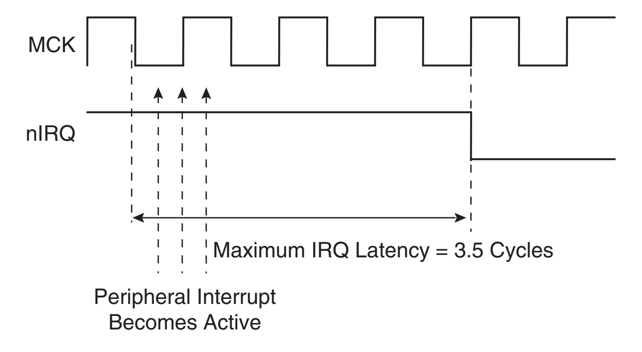
26.7.2.3 Internal Interrupt Edge Triggered Source

Figure 26-8. Internal Interrupt Edge Triggered Source



26.7.2.4 Internal Interrupt Level Sensitive Source

Figure 26-9. Internal Interrupt Level Sensitive Source



26.7.3 Normal Interrupt

26.7.3.1 Priority Controller

An 8-level priority controller drives the nIRQ line of the processor, depending on the interrupt conditions occurring on the interrupt sources 1 to 31 (except for those programmed in Fast Forcing).

Each interrupt source has a programmable priority level of 7 to 0, which is user-definable by writing the PRIOR field of the corresponding AIC_SMR (Source Mode Register). Level 7 is the highest priority and level 0 the lowest.

As soon as an interrupt condition occurs, as defined by the SRCTYPE field of the AIC_SVR (Source Vector Register), the nIRQ line is asserted. As a new interrupt condition might have happened on other interrupt sources since the nIRQ has been asserted, the priority controller determines the current interrupt at the time the AIC_IVR (Interrupt Vector Register) is read. **The read of AIC_IVR is the entry point of the interrupt handling** which allows the AIC to consider that the interrupt has been taken into account by the software.

The current priority level is defined as the priority level of the current interrupt.

If several interrupt sources of equal priority are pending and enabled when the AIC_IVR is read, the interrupt with the lowest interrupt source number is serviced first.

The nIRQ line can be asserted only if an interrupt condition occurs on an interrupt source with a higher priority. If an interrupt condition happens (or is pending) during the interrupt treatment in progress, it is delayed until the software indicates to the AIC the end of the current service by writing the AIC_EOICR (End of Interrupt Command Register). **The write of AIC_EOICR is the exit point of the interrupt handling**.

26.7.3.2 Interrupt Nesting

The priority controller utilizes interrupt nesting in order for the high priority interrupt to be handled during the service of lower priority interrupts. This requires the interrupt service routines of the lower interrupts to re-enable the interrupt at the processor level.

When an interrupt of a higher priority happens during an already occurring interrupt service routine, the nIRQ line is re-asserted. If the interrupt is enabled at the core level, the current execution is interrupted and the new interrupt service routine should read the AIC_IVR. At this time, the current interrupt number and its priority level are pushed into an embedded hardware

stack, so that they are saved and restored when the higher priority interrupt servicing is finished and the AIC_EOICR is written.

The AIC is equipped with an 8-level wide hardware stack in order to support up to eight interrupt nestings pursuant to having eight priority levels.

26.7.3.3 Interrupt Vectoring

The interrupt handler addresses corresponding to each interrupt source can be stored in the registers AIC_SVR1 to AIC_SVR31 (Source Vector Register 1 to 31). When the processor reads AIC_IVR (Interrupt Vector Register), the value written into AIC_SVR corresponding to the current interrupt is returned.

This feature offers a way to branch in one single instruction to the handler corresponding to the current interrupt, as AIC_IVR is mapped at the absolute address 0xFFFF F100 and thus accessible from the ARM interrupt vector at address 0x0000 0018 through the following instruction:

```
LDR  PC,[PC,# -&F20]
```

When the processor executes this instruction, it loads the read value in AIC_IVR in its program counter, thus branching the execution on the correct interrupt handler.

This feature is often not used when the application is based on an operating system (either real time or not). Operating systems often have a single entry point for all the interrupts and the first task performed is to discern the source of the interrupt.

However, it is strongly recommended to port the operating system on AT91 products by supporting the interrupt vectoring. This can be performed by defining all the AIC_SVR of the interrupt source to be handled by the operating system at the address of its interrupt handler. When doing so, the interrupt vectoring permits a critical interrupt to transfer the execution on a specific very fast handler and not onto the operating system's general interrupt handler. This facilitates the support of hard real-time tasks (input/outputs of voice/audio buffers and software peripheral handling) to be handled efficiently and independently of the application running under an operating system.

26.7.3.4 Interrupt Handlers

This section gives an overview of the fast interrupt handling sequence when using the AIC. It is assumed that the programmer understands the architecture of the ARM processor, and especially the processor interrupt modes and the associated status bits.

It is assumed that:

1. The Advanced Interrupt Controller has been programmed, AIC_SVR registers are loaded with corresponding interrupt service routine addresses and interrupts are enabled.
2. The instruction at the ARM interrupt exception vector address is required to work with the vectoring

```
LDR PC, [PC, # -&F20]
```

When nIRQ is asserted, if the bit "I" of CPSR is 0, the sequence is as follows:

1. The CPSR is stored in SPSR_irq, the current value of the Program Counter is loaded in the Interrupt link register (R14_irq) and the Program Counter (R15) is loaded with 0x18. In the following cycle during fetch at address 0x1C, the ARM core adjusts R14_irq, decrementing it by four.
2. The ARM core enters Interrupt mode, if it has not already done so.

3. When the instruction loaded at address 0x18 is executed, the program counter is loaded with the value read in AIC_IVR. Reading the AIC_IVR has the following effects:
   – Sets the current interrupt to be the pending and enabled interrupt with the highest priority. The current level is the priority level of the current interrupt.
   – De-asserts the nIRQ line on the processor. Even if vectoring is not used, AIC_IVR must be read in order to de-assert nIRQ.
   – Automatically clears the interrupt, if it has been programmed to be edge-triggered.
   – Pushes the current level and the current interrupt number on to the stack.
   – Returns the value written in the AIC_SVR corresponding to the current interrupt.
4. The previous step has the effect of branching to the corresponding interrupt service routine. This should start by saving the link register (R14_irq) and SPSR_IRQ. The link register must be decremented by four when it is saved if it is to be restored directly into the program counter at the end of the interrupt. For example, the instruction `SUB PC, LR, #4` may be used.
5. Further interrupts can then be unmasked by clearing the "I" bit in CPSR, allowing re-assertion of the nIRQ to be taken into account by the core. This can happen if an interrupt with a higher priority than the current interrupt occurs.
6. The interrupt handler can then proceed as required, saving the registers that will be used and restoring them at the end. During this phase, an interrupt of higher priority than the current level will restart the sequence from step 1.

Note: If the interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase.

7. The "I" bit in CPSR must be set in order to mask interrupts before exiting to ensure that the interrupt is completed in an orderly manner.
8. The End of Interrupt Command Register (AIC_EOICR) must be written in order to indicate to the AIC that the current interrupt is finished. This causes the current level to be popped from the stack, restoring the previous current level if one exists on the stack. If another interrupt is pending, with lower or equal priority than the old current level but with higher priority than the new current level, the nIRQ line is re-asserted, but the interrupt sequence does not immediately start because the "I" bit is set in the core. SPSR_irq is restored. Finally, the saved value of the link register is restored directly into the PC. This has the effect of returning from the interrupt to whatever was being executed before, and of loading the CPSR with the stored SPSR, masking or unmasking the interrupts depending on the state saved in SPSR_irq.

Note: The "I" bit in SPSR is significant. If it is set, it indicates that the ARM core was on the verge of masking an interrupt when the mask instruction was interrupted. Hence, when SPSR is restored, the mask instruction is completed (interrupt is masked).

26.7.4 Fast Interrupt

26.7.4.1 Fast Interrupt Source

The interrupt source 0 is the only source which can raise a fast interrupt request to the processor except if fast forcing is used. The interrupt source 0 is generally connected to a FIQ pin of the product, either directly or through a PIO Controller.

26.7.4.2 Fast Interrupt Control

The fast interrupt logic of the AIC has no priority controller. The mode of interrupt source 0 is programmed with the AIC_SMR0 and the field PRIOR of this register is not used even if it reads what has been written. The field SRCTYPE of AIC_SMR0 enables programming the

fast interrupt source to be positive-edge triggered or negative-edge triggered or high-level sensitive or low-level sensitive

Writing 0x1 in the AIC_IECR (Interrupt Enable Command Register) and AIC_IDCR (Interrupt Disable Command Register) respectively enables and disables the fast interrupt. The bit 0 of AIC_IMR (Interrupt Mask Register) indicates whether the fast interrupt is enabled or disabled.

26.7.4.3 Fast Interrupt Vectoring

The fast interrupt handler address can be stored in AIC_SVR0 (Source Vector Register 0). The value written into this register is returned when the processor reads AIC_FVR (Fast Vector Register). This offers a way to branch in one single instruction to the interrupt handler, as AIC_FVR is mapped at the absolute address 0xFFFF F104 and thus accessible from the ARM fast interrupt vector at address 0x0000 001C through the following instruction:

```
LDR  PC,[PC,# -&F20]
```

When the processor executes this instruction it loads the value read in AIC_FVR in its program counter, thus branching the execution on the fast interrupt handler. It also automatically performs the clear of the fast interrupt source if it is programmed in edge-triggered mode.

26.7.4.4 Fast Interrupt Handlers

This section gives an overview of the fast interrupt handling sequence when using the AIC. It is assumed that the programmer understands the architecture of the ARM processor, and especially the processor interrupt modes and associated status bits.

Assuming that:

1. The Advanced Interrupt Controller has been programmed, AIC_SVR0 is loaded with the fast interrupt service routine address, and the interrupt source 0 is enabled.
2. The Instruction at address 0x1C (FIQ exception vector address) is required to vector the fast interrupt:

```
LDR PC, [PC, # -&F20]
```

3. The user does not need nested fast interrupts.

When nFIQ is asserted, if the bit "F" of CPSR is 0, the sequence is:

1. The CPSR is stored in SPSR_fiq, the current value of the program counter is loaded in the FIQ link register (R14_FIQ) and the program counter (R15) is loaded with 0x1C. In the following cycle, during fetch at address 0x20, the ARM core adjusts R14_fiq, decrementing it by four.
2. The ARM core enters FIQ mode.
3. When the instruction loaded at address 0x1C is executed, the program counter is loaded with the value read in AIC_FVR. Reading the AIC_FVR has effect of automatically clearing the fast interrupt, if it has been programmed to be edge triggered. In this case only, it de-asserts the nFIQ line on the processor.
4. The previous step enables branching to the corresponding interrupt service routine. It is not necessary to save the link register R14_fiq and SPSR_fiq if nested fast interrupts are not needed.
5. The Interrupt Handler can then proceed as required. It is not necessary to save registers R8 to R13 because FIQ mode has its own dedicated registers and the user R8 to R13 are banked. The other registers, R0 to R7, must be saved before being used, and restored at the end (before the next step). Note that if the fast interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase in order to de-assert the interrupt source 0.

6. Finally, the Link Register R14_fiq is restored into the PC after decrementing it by four (with instruction `SUB PC, LR, #4` for example). This has the effect of returning from the interrupt to whatever was being executed before, loading the CPSR with the SPSR and masking or unmasking the fast interrupt depending on the state saved in the SPSR.

Note: The "F" bit in SPSR is significant. If it is set, it indicates that the ARM core was just about to mask FIQ interrupts when the mask instruction was interrupted. Hence when the SPSR is restored, the interrupted instruction is completed (FIQ is masked).

Another way to handle the fast interrupt is to map the interrupt service routine at the address of the ARM vector 0x1C. This method does not use the vectoring, so that reading AIC_FVR must be performed at the very beginning of the handler operation. However, this method saves the execution of a branch instruction.

26.7.4.5 Fast Forcing

The Fast Forcing feature of the advanced interrupt controller provides redirection of any normal Interrupt source on the fast interrupt controller.

Fast Forcing is enabled or disabled by writing to the Fast Forcing Enable Register (AIC_FFER) and the Fast Forcing Disable Register (AIC_FFDR). Writing to these registers results in an update of the Fast Forcing Status Register (AIC_FFSR) that controls the feature for each internal or external interrupt source.

When Fast Forcing is disabled, the interrupt sources are handled as described in the previous pages.

When Fast Forcing is enabled, the edge/level programming and, in certain cases, edge detection of the interrupt source is still active but the source cannot trigger a normal interrupt to the processor and is not seen by the priority handler.

If the interrupt source is programmed in level-sensitive mode and an active level is sampled, Fast Forcing results in the assertion of the nFIQ line to the core.

If the interrupt source is programmed in edge-triggered mode and an active edge is detected, Fast Forcing results in the assertion of the nFIQ line to the core.

The Fast Forcing feature does not affect the Source 0 pending bit in the Interrupt Pending Register (AIC_IPR).

The Fast Interrupt Vector Register (AIC_FVR) reads the contents of the Source Vector Register 0 (AIC_SVR0), whatever the source of the fast interrupt may be. The read of the FVR does not clear the Source 0 when the fast forcing feature is used and the interrupt source should be cleared by writing to the Interrupt Clear Command Register (AIC_ICCR).

All enabled and pending interrupt sources that have the fast forcing feature enabled and that are programmed in edge-triggered mode must be cleared by writing to the Interrupt Clear Command Register. In doing so, they are cleared independently and thus lost interrupts are prevented.

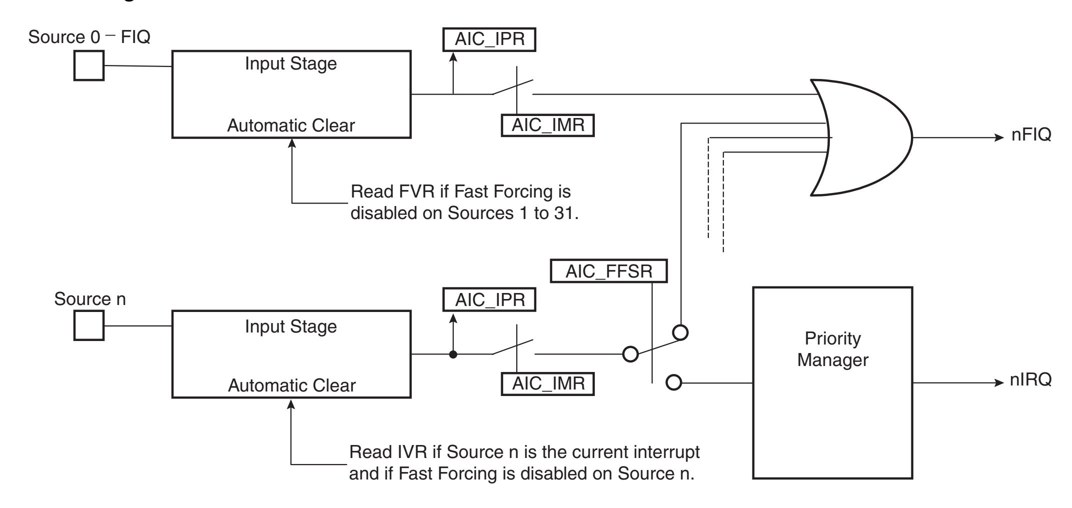
The read of AIC_IVR does not clear the source that has the fast forcing feature enabled.

The source 0, reserved to the fast interrupt, continues operating normally and becomes one of the Fast Interrupt sources.





Figure 26-10. Fast Forcing



26.7.5 Protect Mode

The Protect Mode permits reading the Interrupt Vector Register without performing the associated automatic operations. This is necessary when working with a debug system. When a debugger, working either with a Debug Monitor or the ARM processor's ICE, stops the applications and updates the opened windows, it might read the AIC User Interface and thus the IVR. This has undesirable consequences:

- If an enabled interrupt with a higher priority than the current one is pending, it is stacked.
- If there is no enabled pending interrupt, the spurious vector is returned.

In either case, an End of Interrupt command is necessary to acknowledge and to restore the context of the AIC. This operation is generally not performed by the debug system as the debug system would become strongly intrusive and cause the application to enter an undesired state.

This is avoided by using the Protect Mode. Writing DBGM in AIC_DCR (Debug Control Register) at 0x1 enables the Protect Mode.

When the Protect Mode is enabled, the AIC performs interrupt stacking only when a write access is performed on the AIC_IVR. Therefore, the Interrupt Service Routines must write (arbitrary data) to the AIC_IVR just after reading it. The new context of the AIC, including the value of the Interrupt Status Register (AIC_ISR), is updated with the current interrupt only when AIC_IVR is written.

An AIC_IVR read on its own (e.g., by a debugger), modifies neither the AIC context nor the AIC_ISR. Extra AIC_IVR reads perform the same operations. However, it is recommended to not stop the processor between the read and the write of AIC_IVR of the interrupt service routine to make sure the debugger does not modify the AIC context.

To summarize, in normal operating mode, the read of AIC_IVR performs the following operations within the AIC:

1. Calculates active interrupt (higher than current or spurious).
2. Determines and returns the vector of the active interrupt.
3. Memorizes the interrupt.
4. Pushes the current priority level onto the internal stack.

5. Acknowledges the interrupt.

However, while the Protect Mode is activated, only operations 1 to 3 are performed when AIC_IVR is read. Operations 4 and 5 are only performed by the AIC when AIC_IVR is written.

Software that has been written and debugged using the Protect Mode runs correctly in Normal Mode without modification. However, in Normal Mode the AIC_IVR write has no effect and can be removed to optimize the code.

26.7.6 Spurious Interrupt

The Advanced Interrupt Controller features protection against spurious interrupts. A spurious interrupt is defined as being the assertion of an interrupt source long enough for the AIC to assert the nIRQ, but no longer present when AIC_IVR is read. This is most prone to occur when:

- An external interrupt source is programmed in level-sensitive mode and an active level occurs for only a short time.
- An internal interrupt source is programmed in level sensitive and the output signal of the corresponding embedded peripheral is activated for a short time. (As in the case for the Watchdog.)
- An interrupt occurs just a few cycles before the software begins to mask it, thus resulting in a pulse on the interrupt source.

The AIC detects a spurious interrupt at the time the AIC_IVR is read while no enabled interrupt source is pending. When this happens, the AIC returns the value stored by the programmer in AIC_SPU (Spurious Vector Register). The programmer must store the address of a spurious interrupt handler in AIC_SPU as part of the application, to enable an as fast as possible return to the normal execution flow. This handler writes in AIC_EOICR and performs a return from interrupt.

26.7.7 General Interrupt Mask

The AIC features a General Interrupt Mask bit to prevent interrupts from reaching the processor. Both the nIRQ and the nFIQ lines are driven to their inactive state if the bit GMSK in AIC_DCR (Debug Control Register) is set. However, this mask does not prevent waking up the processor if it has entered Idle Mode. This function facilitates synchronizing the processor on a next event and, as soon as the event occurs, performs subsequent operations without having to handle an interrupt. It is strongly recommended to use this mask with caution.

26.8 Advanced Interrupt Controller (AIC) User Interface

26.8.1 Base Address

The AIC is mapped at the address **0xFFFF F000**. It has a total 4-Kbyte addressing space. This permits the vectoring feature, as the PC-relative load/store instructions of the ARM processor support only a ± 4-Kbyte offset.

26.8.2 Register Mapping

Table 26-2. Register Mapping

Offset	Register	Name	Access	Reset Value
0000	Source Mode Register 0	AIC_SMR0	Read/Write	0x0
0x04	Source Mode Register 1	AIC_SMR1	Read/Write	0x0
---	---	---	---	---
0x7C	Source Mode Register 31	AIC_SMR31	Read/Write	0x0
0x80	Source Vector Register 0	AIC_SVR0	Read/Write	0x0
0x84	Source Vector Register 1	AIC_SVR1	Read/Write	0x0
---	---	---	---	---
0xFC	Source Vector Register 31	AIC_SVR31	Read/Write	0x0
0x100	Interrupt Vector Register	AIC_IVR	Read-only	0x0
0x104	Fast Interrupt Vector Register	AIC_FVR	Read-only	0x0
0x108	Interrupt Status Register	AIC_ISR	Read-only	0x0
0x10C	Interrupt Pending Register	AIC_IPR	Read-only	0x0[1]
0x110	Interrupt Mask Register	AIC_IMR	Read-only	0x0
0x114	Core Interrupt Status Register	AIC_CISR	Read-only	0x0
0x118	Reserved	---	---	---
0x11C	Reserved	---	---	---
0x120	Interrupt Enable Command Register	AIC_IECR	Write-only	---
0x124	Interrupt Disable Command Register	AIC_IDCR	Write-only	---
0x128	Interrupt Clear Command Register	AIC_ICCR	Write-only	---
0x12C	Interrupt Set Command Register	AIC_ISCR	Write-only	---
0x130	End of Interrupt Command Register	AIC_EOICR	Write-only	---
0x134	Spurious Interrupt Vector Register	AIC_SPU	Read/Write	0x0
0x138	Debug Control Register	AIC_DCR	Read/Write	0x0
0x13C	Reserved	---	---	---
0x140	Fast Forcing Enable Register	AIC_FFER	Write-only	---
0x144	Fast Forcing Disable Register	AIC_FFDR	Write-only	---
0x148	Fast Forcing Status Register	AIC_FFSR	Read-only	0x0

Note: 1. The reset value of this register depends on the level of the external interrupt source. All other sources are cleared at reset, thus not pending.

26.8.3 AIC Source Mode Register

Register Name: AIC_SMR0..AIC_SMR31

Access Type: Read/Write

Reset Value: 0x0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	SRCTYPE		–	–	PRIOR		

- **PRIOR: Priority Level**

Programs the priority level for all sources except FIQ source (source 0).

The priority level can be between 0 (lowest) and 7 (highest).

The priority level is not used for the FIQ in the related SMR register AIC_SMRx.

- **SRCTYPE: Interrupt Source Type**

The active level or edge is not programmable for the internal interrupt sources.

SRCTYPE		Internal Interrupt Sources	External Interrupt Sources
0	0	High level Sensitive	Low level Sensitive
0	1	Positive edge triggered	Negative edge triggered
1	0	High level Sensitive	High level Sensitive
1	1	Positive edge triggered	Positive edge triggered

26.8.4 AIC Source Vector Register

Register Name: AIC_SVR0..AIC_SVR31

Access Type: Read/Write

Reset Value: 0x0

31	30	29	28	27	26	25	24
VECTOR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
VECTOR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
VECTOR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VECTOR							

- **VECTOR: Source Vector**

The user may store in these registers the addresses of the corresponding handler for each interrupt source.

26.8.5 AIC Interrupt Vector Register

Register Name: AIC_IVR

Access Type: Read-only

Reset Value: 0

31	30	29	28	27	26	25	24
IRQV							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
IRQV							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
IRQV							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IRQV							

- **IRQV: Interrupt Vector Register**

The Interrupt Vector Register contains the vector programmed by the user in the Source Vector Register corresponding to the current interrupt.

The Source Vector Register is indexed using the current interrupt number when the Interrupt Vector Register is read.

When there is no current interrupt, the Interrupt Vector Register reads the value stored in AIC_SPU.

26.8.6 AIC FIQ Vector Register

Register Name: AIC_FVR

Access Type: Read-only

Reset Value: 0

31	30	29	28	27	26	25	24
FIQV							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
FIQV							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FIQV							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FIQV							

- **FIQV: FIQ Vector Register**

The FIQ Vector Register contains the vector programmed by the user in the Source Vector Register 0. When there is no fast interrupt, the Fast Interrupt Vector Register reads the value stored in AIC_SPU.

26.8.7 AIC Interrupt Status Register

Register Name: AIC_ISR

Access Type: Read-only

Reset Value: 0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	IRQID				

- **IRQID: Current Interrupt Identifier**

The Interrupt Status Register returns the current interrupt source number.

26.8.8 AIC Interrupt Pending Register

Register Name: AIC_IPR

Access Type: Read-only

Reset Value: 0

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Pending**

0 = Corresponding interrupt is not pending.

1 = Corresponding interrupt is pending.

26.8.9 AIC Interrupt Mask Register

Register Name: AIC_IMR

Access Type: Read-only

Reset Value: 0

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Mask**

0 = Corresponding interrupt is disabled.

1 = Corresponding interrupt is enabled.

26.8.10 AIC Core Interrupt Status Register

Register Name: AIC_CISR

Access Type: Read-only

Reset Value: 0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	NIRQ	NIFQ

- **NFIQ: NFIQ Status**

0 = nFIQ line is deactivated.

1 = nFIQ line is active.

- **NIRQ: NIRQ Status**

0 = nIRQ line is deactivated.

1 = nIRQ line is active.

26.8.11 AIC Interrupt Enable Command Register

Register Name: AIC_IECR

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID3: Interrupt Enable**

0 = No effect.

1 = Enables corresponding interrupt.

26.8.12 AIC Interrupt Disable Command Register

Register Name: AIC_IDCR

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Disable**

0 = No effect.

1 = Disables corresponding interrupt.

26.8.13 AIC Interrupt Clear Command Register

Register Name: AIC_ICCR

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Clear**

0 = No effect.

1 = Clears corresponding interrupt.

26.8.14 AIC Interrupt Set Command Register

Register Name: AIC_ISCR

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Set**

0 = No effect.

1 = Sets corresponding interrupt.

26.8.15 AIC End of Interrupt Command Register

Register Name: AIC_EOICR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

The End of Interrupt Command Register is used by the interrupt routine to indicate that the interrupt treatment is complete. Any value can be written because it is only necessary to make a write to this register location to signal the end of interrupt treatment.

26.8.16 AIC Spurious Interrupt Vector Register

Register Name: AIC_SPU

Access Type: Read/Write

Reset Value: 0

31	30	29	28	27	26	25	24
SIQV							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SIQV							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SIQV							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SIQV							

- **SIQV: Spurious Interrupt Vector Register**

The user may store the address of a spurious interrupt handler in this register. The written value is returned in AIC_IVR in case of a spurious interrupt and in AIC_FVR in case of a spurious fast interrupt.

26.8.17 AIC Debug Control Register

Register Name: AIC_DEBUG

Access Type: Read/Write

Reset Value: 0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	GMSK	PROT

- **PROT: Protection Mode**

0 = The Protection Mode is disabled.

1 = The Protection Mode is enabled.

- **GMSK: General Mask**

0 = The nIRQ and nFIQ lines are normally controlled by the AIC.

1 = The nIRQ and nFIQ lines are tied to their inactive state.

26.8.18 AIC Fast Forcing Enable Register

Register Name: AIC_FFER

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2-PID31: Fast Forcing Enable**

0 = No effect.

1 = Enables the fast forcing feature on the corresponding interrupt.

26.8.19 AIC Fast Forcing Disable Register

Register Name: AIC_FFDR

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2-PID31: Fast Forcing Disable**

0 = No effect.

1 = Disables the Fast Forcing feature on the corresponding interrupt.

26.8.20 AIC Fast Forcing Status Register

Register Name: AIC_FFSR

Access Type: Read-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2-PID31: Fast Forcing Status**

0 = The Fast Forcing feature is disabled on the corresponding interrupt.

1 = The Fast Forcing feature is enabled on the corresponding interrupt.



27. Debug Unit (DBGU)

27.1 Description

The Debug Unit provides a single entry point from the processor for access to all the debug capabilities of Atmel's ARM-based systems.

The Debug Unit features a two-pin UART that can be used for several debug and trace purposes and offers an ideal medium for in-situ programming solutions and debug monitor communications. Moreover, the association with two peripheral data controller channels permits packet handling for these tasks with processor time reduced to a minimum.

The Debug Unit also makes the Debug Communication Channel (DCC) signals provided by the In-circuit Emulator of the ARM processor visible to the software. These signals indicate the status of the DCC read and write registers and generate an interrupt to the ARM processor, making possible the handling of the DCC under interrupt control.

Chip Identifier registers permit recognition of the device and its revision. These registers inform as to the sizes and types of the on-chip memories, as well as the set of embedded peripherals.

Finally, the Debug Unit features a Force NTRST capability that enables the software to decide whether to prevent access to the system via the In-circuit Emulator. This permits protection of the code, stored in ROM.

27.2 Block Diagram

Figure 27-1. Debug Unit Functional Block Diagram

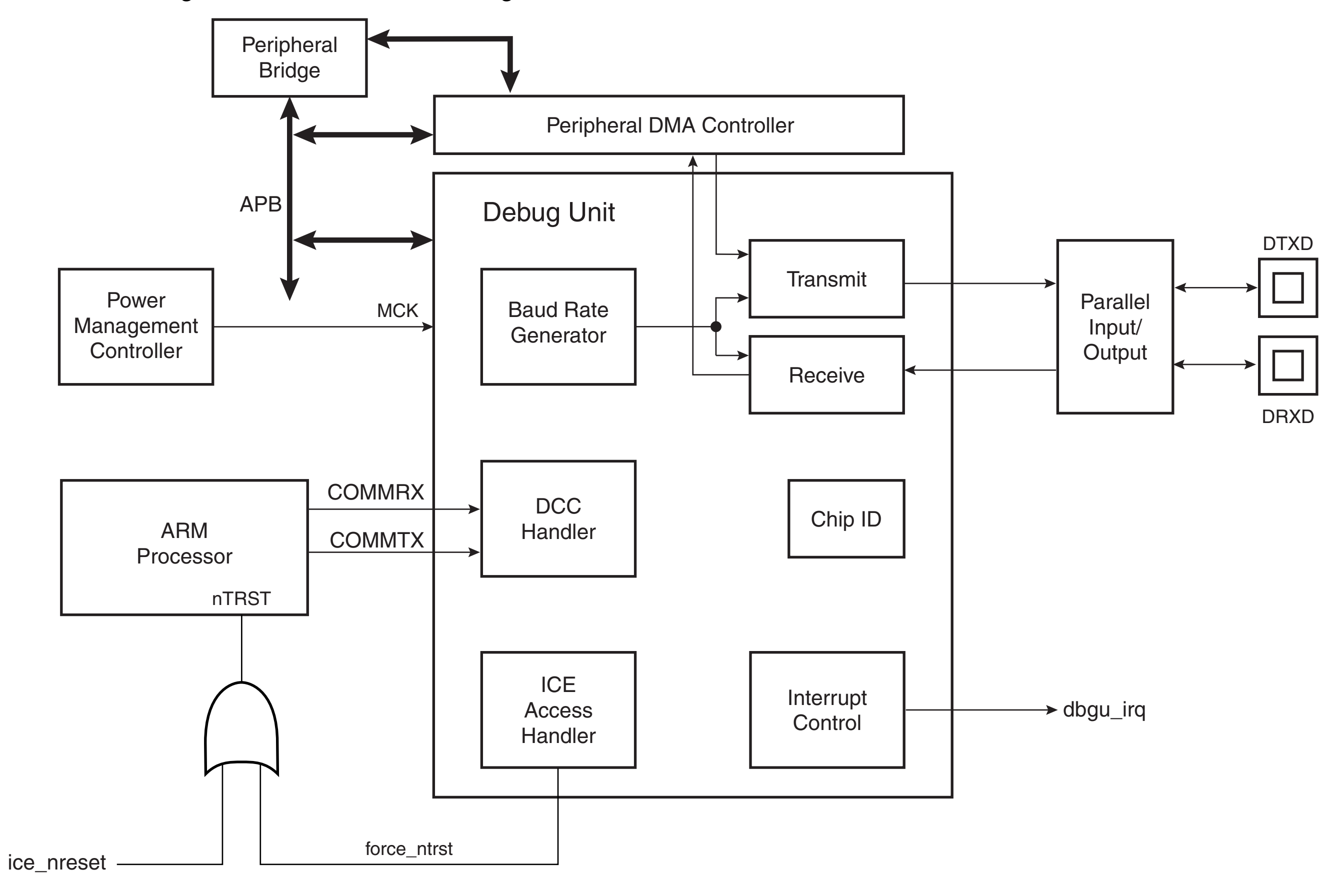
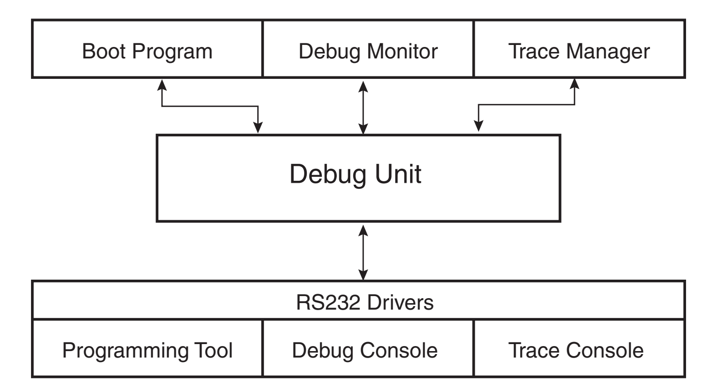


Table 27-1. Debug Unit Pin Description

Pin Name	Description	Type
DRXD	Debug Receive Data	Input
DTXD	Debug Transmit Data	Output

Figure 27-2. Debug Unit Application Example



27.3 Product Dependencies

27.3.1 I/O Lines

Depending on product integration, the Debug Unit pins may be multiplexed with PIO lines. In this case, the programmer must first configure the corresponding PIO Controller to enable I/O lines operations of the Debug Unit.

27.3.2 Power Management

Depending on product integration, the Debug Unit clock may be controllable through the Power Management Controller. In this case, the programmer must first configure the PMC to enable the Debug Unit clock. Usually, the peripheral identifier used for this purpose is 1.

27.3.3 Interrupt Source

Depending on product integration, the Debug Unit interrupt line is connected to one of the interrupt sources of the Advanced Interrupt Controller. Interrupt handling requires programming of the AIC before configuring the Debug Unit. Usually, the Debug Unit interrupt line connects to the interrupt source 1 of the AIC, which may be shared with the real-time clock, the system timer interrupt lines and other system peripheral interrupts, as shown in Figure 27-1. This sharing requires the programmer to determine the source of the interrupt when the source 1 is triggered.

27.4 UART Operations

The Debug Unit operates as a UART, (asynchronous mode only) and supports only 8-bit character handling (with parity). It has no clock pin.

The Debug Unit's UART is made up of a receiver and a transmitter that operate independently, and a common baud rate generator. Receiver timeout and transmitter time guard are not implemented. However, all the implemented features are compatible with those of a standard USART.

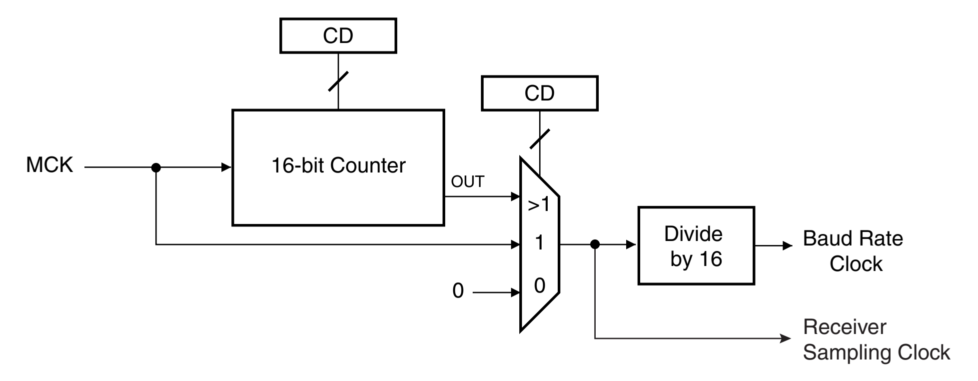
27.4.1 Baud Rate Generator

The baud rate generator provides the bit period clock named baud rate clock to both the receiver and the transmitter.

The baud rate clock is the master clock divided by 16 times the value (CD) written in DBGU_BRGR (Baud Rate Generator Register). If DBGU_BRGR is set to 0, the baud rate clock is disabled and the Debug Unit's UART remains inactive. The maximum allowable baud rate is Master Clock divided by 16. The minimum allowable baud rate is Master Clock divided by (16 x 65536).

$$\text{Baud Rate} = \frac{MCK}{16 \times CD}$$

Figure 27-3. Baud Rate Generator



27.4.2 Receiver

27.4.2.1 Receiver Reset, Enable and Disable

After device reset, the Debug Unit receiver is disabled and must be enabled before being used. The receiver can be enabled by writing the control register DBGU_CR with the bit RXEN at 1. At this command, the receiver starts looking for a start bit.

The programmer can disable the receiver by writing DBGU_CR with the bit RXDIS at 1. If the receiver is waiting for a start bit, it is immediately stopped. However, if the receiver has already detected a start bit and is receiving the data, it waits for the stop bit before actually stopping its operation.

The programmer can also put the receiver in its reset state by writing DBGU_CR with the bit RSTRX at 1. In doing so, the receiver immediately stops its current operations and is disabled, whatever its current state. If RSTRX is applied when data is being processed, this data is lost.

27.4.2.2 Start Detection and Data Sampling

The Debug Unit only supports asynchronous operations, and this affects only its receiver. The Debug Unit receiver detects the start of a received character by sampling the DRXD signal until it detects a valid start bit. A low level (space) on DRXD is interpreted as a valid start bit if it is detected for more than 7 cycles of the sampling clock, which is 16 times the baud rate. Hence, a space that is longer than 7/16 of the bit period is detected as a valid start bit. A space which is 7/16 of a bit period or shorter is ignored and the receiver continues to wait for a valid start bit.

When a valid start bit has been detected, the receiver samples the DRXD at the theoretical midpoint of each bit. It is assumed that each bit lasts 16 cycles of the sampling clock (1-bit period) so the bit sampling point is eight cycles (0.5-bit period) after the start of the bit. The first sampling point is therefore 24 cycles (1.5-bit periods) after the falling edge of the start bit was detected.

Each subsequent bit is sampled 16 cycles (1-bit period) after the previous one.

Figure 27-4. Start Bit Detection

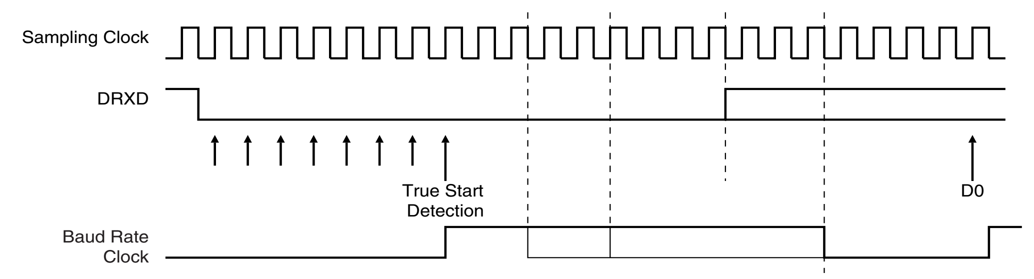
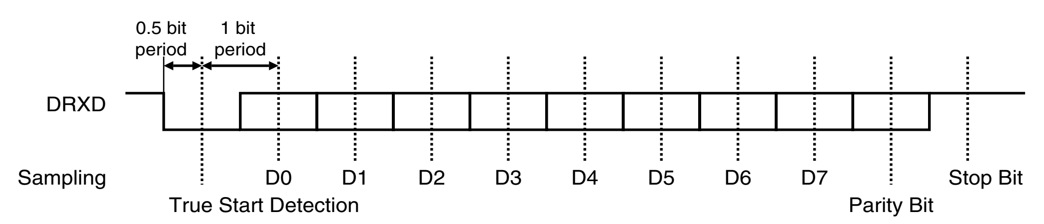


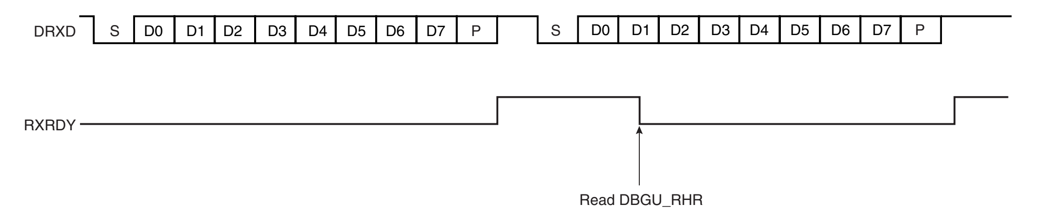
Figure 27-5. Character Reception



27.4.2.3 *Receiver Ready*

When a complete character is received, it is transferred to the DBGU_RHR and the RXRDY status bit in DBGU_SR (Status Register) is set. The bit RXRDY is automatically cleared when the receive holding register DBGU_RHR is read.

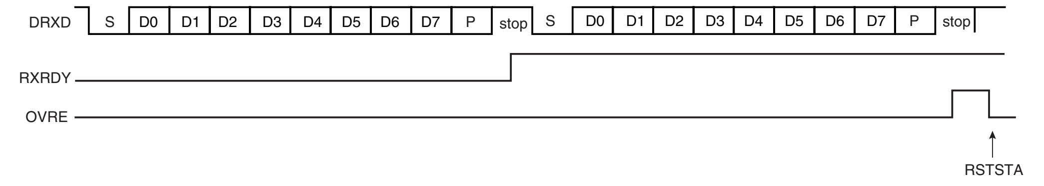
Figure 27-6. Receiver Ready



27.4.2.4 *Receiver Overrun*

If DBGU_RHR has not been read by the software (or the Peripheral Data Controller) since the last transfer, the RXRDY bit is still set and a new character is received, the OVRE status bit in DBGU_SR is set. OVRE is cleared when the software writes the control register DBGU_CR with the bit RSTSTA (Reset Status) at 1.

Figure 27-7. Receiver Overrun

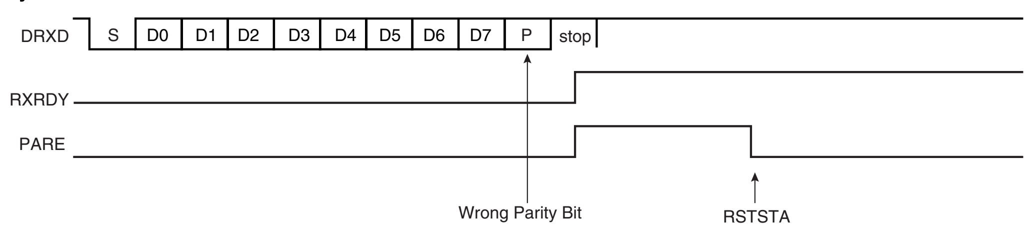


27.4.2.5 *Parity Error*

Each time a character is received, the receiver calculates the parity of the received data bits, in accordance with the field PAR in DBGU_MR. It then compares the result with the received

parity bit. If different, the parity error bit PARE in DBGU_SR is set at the same time the RXRDY is set. The parity bit is cleared when the control register DBGU_CR is written with the bit RSTSTA (Reset Status) at 1. If a new character is received before the reset status command is written, the PARE bit remains at 1.

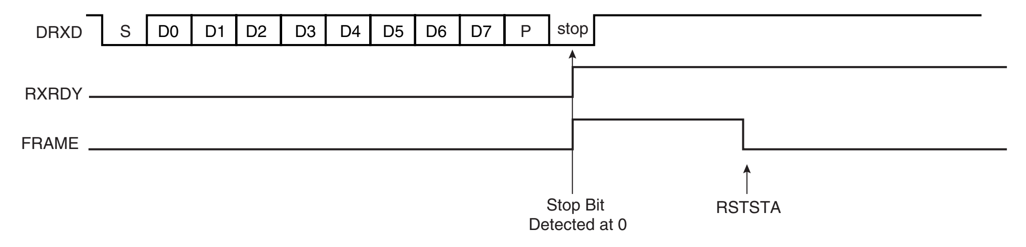
Figure 27-8. Parity Error



27.4.2.6 Receiver Framing Error

When a start bit is detected, it generates a character reception when all the data bits have been sampled. The stop bit is also sampled and when it is detected at 0, the FRAME (Framing Error) bit in DBGU_SR is set at the same time the RXRDY bit is set. The bit FRAME remains high until the control register DBGU_CR is written with the bit RSTSTA at 1.

Figure 27-9. Receiver Framing Error



27.4.3 Transmitter

27.4.3.1 Transmitter Reset, Enable and Disable

After device reset, the Debug Unit transmitter is disabled and it must be enabled before being used. The transmitter is enabled by writing the control register DBGU_CR with the bit TXEN at 1. From this command, the transmitter waits for a character to be written in the Transmit Holding Register DBGU_THR before actually starting the transmission.

The programmer can disable the transmitter by writing DBGU_CR with the bit TXDIS at 1. If the transmitter is not operating, it is immediately stopped. However, if a character is being processed into the Shift Register and/or a character has been written in the Transmit Holding Register, the characters are completed before the transmitter is actually stopped.

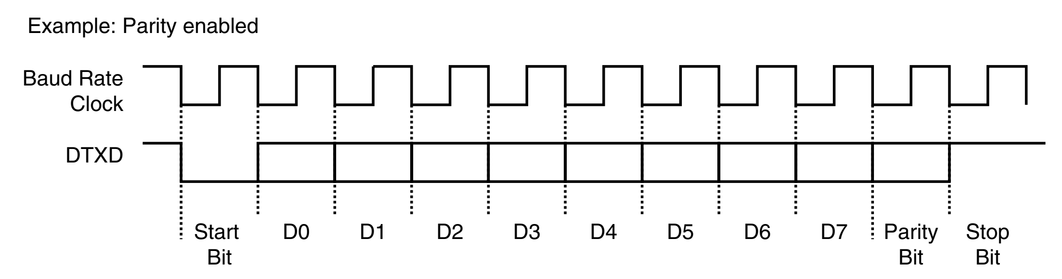
The programmer can also put the transmitter in its reset state by writing the DBGU_CR with the bit RSTTX at 1. This immediately stops the transmitter, whether or not it is processing characters.

27.4.3.2 Transmit Format

The Debug Unit transmitter drives the pin DTXD at the baud rate clock speed. The line is driven depending on the format defined in the Mode Register and the data stored in the Shift Register. One start bit at level 0, then the 8 data bits, from the lowest to the highest bit, one optional parity bit and one stop bit at 1 are consecutively shifted out as shown on the following

figure. The field PARE in the mode register DBGU_MR defines whether or not a parity bit is shifted out. When a parity bit is enabled, it can be selected between an odd parity, an even parity, or a fixed space or mark bit.

Figure 27-10. Character Transmission

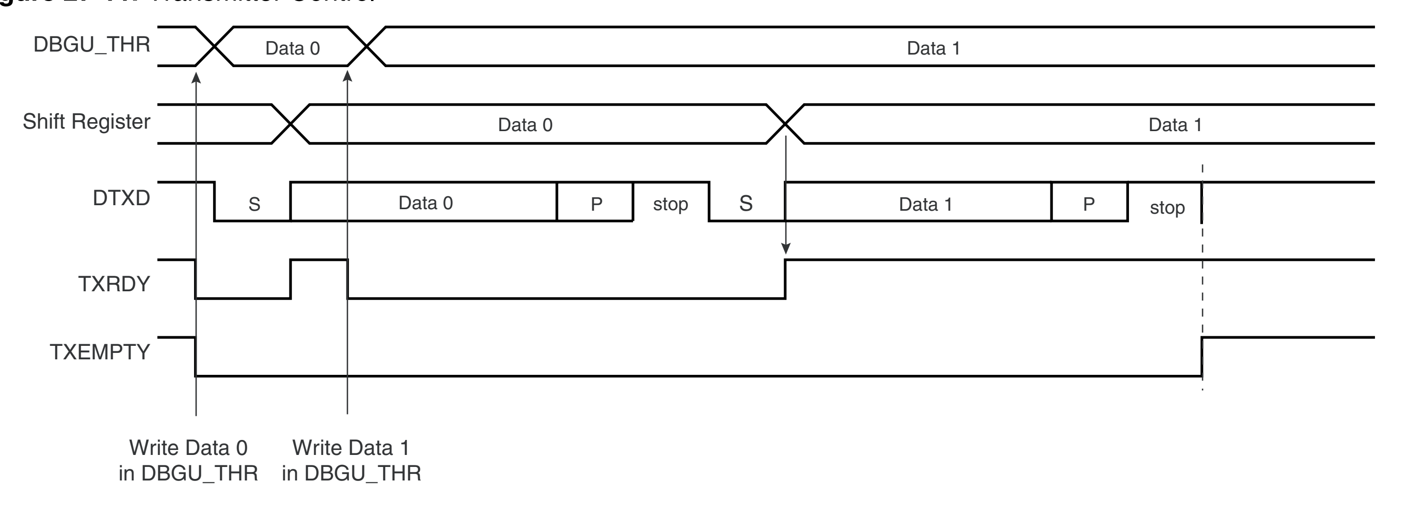


27.4.3.3 Transmitter Control

When the transmitter is enabled, the bit TXRDY (Transmitter Ready) is set in the status register DBGU_SR. The transmission starts when the programmer writes in the Transmit Holding Register DBGU_THR, and after the written character is transferred from DBGU_THR to the Shift Register. The bit TXRDY remains high until a second character is written in DBGU_THR. As soon as the first character is completed, the last character written in DBGU_THR is transferred into the shift register and TXRDY rises again, showing that the holding register is empty.

When both the Shift Register and the DBGU_THR are empty, i.e., all the characters written in DBGU_THR have been processed, the bit TXEMPTY rises after the last stop bit has been completed.

Figure 27-11. Transmitter Control



27.4.4 Peripheral Data Controller

Both the receiver and the transmitter of the Debug Unit's UART are generally connected to a Peripheral Data Controller (PDC) channel.

The peripheral data controller channels are programmed via registers that are mapped within the Debug Unit user interface from the offset 0x100. The status bits are reported in the Debug Unit status register DBGU_SR and can generate an interrupt.

The RXRDY bit triggers the PDC channel data transfer of the receiver. This results in a read of the data in DBGU_RHR. The TXRDY bit triggers the PDC channel data transfer of the transmitter. This results in a write of a data in DBGU_THR.

27.4.5 Test Modes

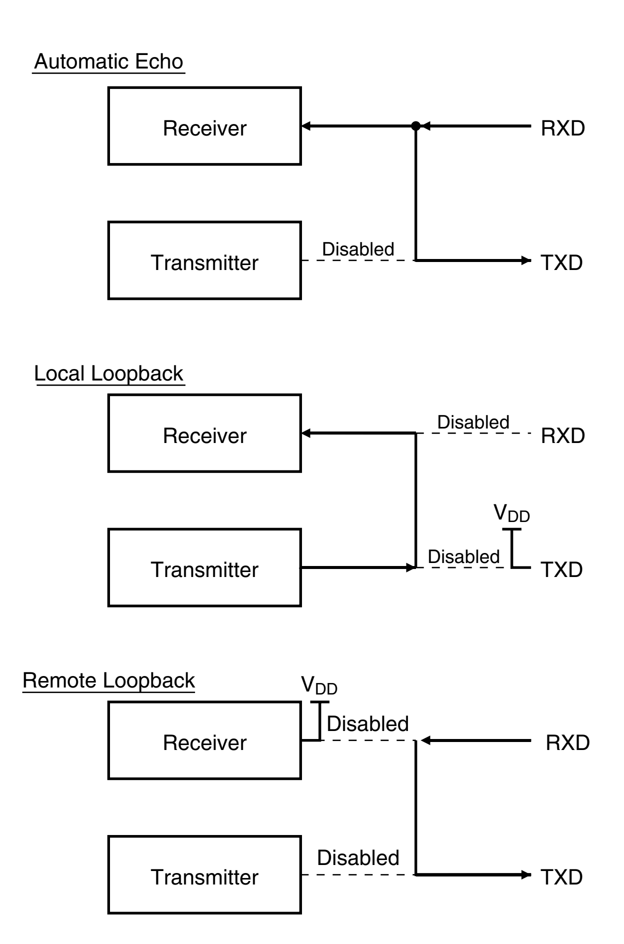
The Debug Unit supports three tests modes. These modes of operation are programmed by using the field CHMODE (Channel Mode) in the mode register DBGU_MR.

The Automatic Echo mode allows bit-by-bit retransmission. When a bit is received on the DRXD line, it is sent to the DTXD line. The transmitter operates normally, but has no effect on the DTXD line.

The Local Loopback mode allows the transmitted characters to be received. DTXD and DRXD pins are not used and the output of the transmitter is internally connected to the input of the receiver. The DRXD pin level has no effect and the DTXD line is held high, as in idle state.

The Remote Loopback mode directly connects the DRXD pin to the DTXD line. The transmitter and the receiver are disabled and have no effect. This mode allows a bit-by-bit retransmission.

Figure 27-12. Test Modes



27.4.6 Debug Communication Channel Support

The Debug Unit handles the signals COMMRX and COMMTX that come from the Debug Communication Channel of the ARM Processor and are driven by the In-circuit Emulator.



The Debug Communication Channel contains two registers that are accessible through the ICE Breaker on the JTAG side and through the coprocessor 0 on the ARM Processor side.

As a reminder, the following instructions are used to read and write the Debug Communication Channel:

```
MRC    p14, 0, Rd, c1, c0, 0
```

Returns the debug communication data read register into Rd

```
MCR    p14, 0, Rd, c1, c0, 0
```

Writes the value in Rd to the debug communication data write register.

The bits COMMRX and COMMTX, which indicate, respectively, that the read register has been written by the debugger but not yet read by the processor, and that the write register has been written by the processor and not yet read by the debugger, are wired on the two highest bits of the status register DBGU_SR. These bits can generate an interrupt. This feature permits handling under interrupt a debug link between a debug monitor running on the target system and a debugger.

27.4.7 Chip Identifier

The Debug Unit features two chip identifier registers, DBGU_CIDR (Chip ID Register) and DBGU_EXID (Extension ID). Both registers contain a hard-wired value that is read-only. The first register contains the following fields:

- EXT - shows the use of the extension identifier register
- NVPTYP and NVPSIZ - identifies the type of embedded non-volatile memory and its size
- ARCH - identifies the set of embedded peripheral
- SRAMSIZ - indicates the size of the embedded SRAM
- EPROC - indicates the embedded ARM processor
- VERSION - gives the revision of the silicon

The second register is device-dependent and reads 0 if the bit EXT is 0.

27.4.8 ICE Access Prevention

The Debug Unit allows blockage of access to the system through the ARM processor's ICE interface. This feature is implemented via the register Force NTRST (DBGU_FNR), that allows assertion of the NTRST signal of the ICE Interface. Writing the bit FNTRST (Force NTRST) to 1 in this register prevents any activity on the TAP controller.

On standard devices, the bit FNTRST resets to 0 and thus does not prevent ICE access.

This feature is especially useful on custom ROM devices for customers who do not want their on-chip code to be visible.

27.5 Debug Unit (DBGU) User Interface

Table 27-2. Debug Unit Memory Map

Offset	Register	Name	Access	Reset Value
0x0000	Control Register	DBGU_CR	Write-only	–
0x0004	Mode Register	DBGU_MR	Read/Write	0x0
0x0008	Interrupt Enable Register	DBGU_IER	Write-only	–
0x000C	Interrupt Disable Register	DBGU_IDR	Write-only	–
0x0010	Interrupt Mask Register	DBGU_IMR	Read-only	0x0
0x0014	Status Register	DBGU_SR	Read-only	–
0x0018	Receive Holding Register	DBGU_RHR	Read-only	0x0
0x001C	Transmit Holding Register	DBGU_THR	Write-only	–
0x0020	Baud Rate Generator Register	DBGU_BRGR	Read/Write	0x0
0x0024 - 0x003C	Reserved	–	–	–
0x0040	Chip ID Register	DBGU_CIDR	Read-only	–
0x0044	Chip ID Extension Register	DBGU_EXID	Read-only	–
0x0048	Force NTRST Register	DBGU_FNR	Read/Write	0x0
0x004C - 0x00FC	Reserved	–	–	–
0x0100 - 0x0124	PDC Area	–	–	–

27.5.1 Debug Unit Control Register

Name: DBGU_CR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	RSTSTA

7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

- **RSTRX: Reset Receiver**

0 = No effect.

1 = The receiver logic is reset and disabled. If a character is being received, the reception is aborted.

- **RSTTX: Reset Transmitter**

0 = No effect.

1 = The transmitter logic is reset and disabled. If a character is being transmitted, the transmission is aborted.

- **RXEN: Receiver Enable**

0 = No effect.

1 = The receiver is enabled if RXDIS is 0.

- **RXDIS: Receiver Disable**

0 = No effect.

1 = The receiver is disabled. If a character is being processed and RSTRX is not set, the character is completed before the receiver is stopped.

- **TXEN: Transmitter Enable**

0 = No effect.

1 = The transmitter is enabled if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0 = No effect.

1 = The transmitter is disabled. If a character is being processed and a character has been written the DBGU_THR and RSTTX is not set, both characters are completed before the transmitter is stopped.

- **RSTSTA: Reset Status Bits**

0 = No effect.

1 = Resets the status bits PARE, FRAME and OVRE in the DBGU_SR.

27.5.2 Debug Unit Mode Register

Name: DBGU_MR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
CHMODE		–	–	PAR			–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **PAR: Parity Type**

PAR			Parity Type
0	0	0	Even parity
0	0	1	Odd parity
0	1	0	Space: parity forced to 0
0	1	1	Mark: parity forced to 1
1	x	x	No parity

- **CHMODE: Channel Mode**

CHMODE		Mode Description
0	0	Normal Mode
0	1	Automatic Echo
1	0	Local Loopback
1	1	Remote Loopback

27.5.3 Debug Unit Interrupt Enable Register

Name: DBGU_IER

Access Type: Write-only

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: Enable RXRDY Interrupt**
- **TXRDY: Enable TXRDY Interrupt**
- **ENDRX: Enable End of Receive Transfer Interrupt**
- **ENDTX: Enable End of Transmit Interrupt**
- **OVRE: Enable Overrun Error Interrupt**
- **FRAME: Enable Framing Error Interrupt**
- **PARE: Enable Parity Error Interrupt**
- **TXEMPTY: Enable TXEMPTY Interrupt**
- **TXBUFE: Enable Buffer Empty Interrupt**
- **RXBUFF: Enable Buffer Full Interrupt**
- **COMMTX: Enable COMMTX (from ARM) Interrupt**
- **COMMRX: Enable COMMRX (from ARM) Interrupt**

0 = No effect.

1 = Enables the corresponding interrupt.



27.5.4 Debug Unit Interrupt Disable Register

Name: DBGU_IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: Disable RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **ENDRX: Disable End of Receive Transfer Interrupt**
- **ENDTX: Disable End of Transmit Interrupt**
- **OVRE: Disable Overrun Error Interrupt**
- **FRAME: Disable Framing Error Interrupt**
- **PARE: Disable Parity Error Interrupt**
- **TXEMPTY: Disable TXEMPTY Interrupt**
- **TXBUFE: Disable Buffer Empty Interrupt**
- **RXBUFF: Disable Buffer Full Interrupt**
- **COMMTX: Disable COMMTX (from ARM) Interrupt**
- **COMMRX: Disable COMMRX (from ARM) Interrupt**

0 = No effect.

1 = Disables the corresponding interrupt.

27.5.5 Debug Unit Interrupt Mask Register

Name: DBGU_IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXDY

- **RXRDY: Mask RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **ENDRX: Mask End of Receive Transfer Interrupt**
- **ENDTX: Mask End of Transmit Interrupt**
- **OVRE: Mask Overrun Error Interrupt**
- **FRAME: Mask Framing Error Interrupt**
- **PARE: Mask Parity Error Interrupt**
- **TXEMPTY: Mask TXEMPTY Interrupt**
- **TXBUFE: Mask TXBUFE Interrupt**
- **RXBUFF: Mask RXBUFF Interrupt**
- **COMMTX: Mask COMMTX Interrupt**
- **COMMRX: Mask COMMRX Interrupt**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

27.5.6 Debug Unit Status Register

Name: DBGU_SR

Access Type: Read-only

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: Receiver Ready**

0 = No character has been received since the last read of the DBGU_RHR or the receiver is disabled.

1 = At least one complete character has been received, transferred to DBGU_RHR and not yet read.

- **TXRDY: Transmitter Ready**

0 = A character has been written to DBGU_THR and not yet transferred to the Shift Register, or the transmitter is disabled.

1 = There is no character written to DBGU_THR not yet transferred to the Shift Register.

- **ENDRX: End of Receiver Transfer**

0 = The End of Transfer signal from the receiver Peripheral Data Controller channel is inactive.

1 = The End of Transfer signal from the receiver Peripheral Data Controller channel is active.

- **ENDTX: End of Transmitter Transfer**

0 = The End of Transfer signal from the transmitter Peripheral Data Controller channel is inactive.

1 = The End of Transfer signal from the transmitter Peripheral Data Controller channel is active.

- **OVRE: Overrun Error**

0 = No overrun error has occurred since the last RSTSTA.

1 = At least one overrun error has occurred since the last RSTSTA.

- **FRAME: Framing Error**

0 = No framing error has occurred since the last RSTSTA.

1 = At least one framing error has occurred since the last RSTSTA.

- **PARE: Parity Error**

0 = No parity error has occurred since the last RSTSTA.

1 = At least one parity error has occurred since the last RSTSTA.

- **TXEMPTY: Transmitter Empty**

0 = There are characters in DBGU_THR, or characters being processed by the transmitter, or the transmitter is disabled.

1 = There are no characters in DBGU_THR and there are no characters being processed by the transmitter.

- **TXBUFE: Transmission Buffer Empty**

0 = The buffer empty signal from the transmitter PDC channel is inactive.

1 = The buffer empty signal from the transmitter PDC channel is active.

- **RXBUFF: Receive Buffer Full**

0 = The buffer full signal from the receiver PDC channel is inactive.

1 = The buffer full signal from the receiver PDC channel is active.

- **COMMTX: Debug Communication Channel Write Status**

0 = COMMTX from the ARM processor is inactive.

1 = COMMTX from the ARM processor is active.

- **COMMRX: Debug Communication Channel Read Status**

0 = COMMRX from the ARM processor is inactive.

1 = COMMRX from the ARM processor is active.

27.5.7 Debug Unit Receiver Holding Register

Name: DBGU_RHR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXCHR							

- **RXCHR: Received Character**

Last received character if RXRDY is set.

27.5.8 Debug Unit Transmit Holding Register

Name: DBGU_THR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXCHR							

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set.

27.5.9 Debug Unit Baud Rate Generator Register

Name: DBGU_BRGR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
CD							

7	6	5	4	3	2	1	0
CD							

- **CD: Clock Divisor**

CD	Baud Rate Clock
0	Disabled
1	MCK
2 to 65535	MCK / (CD x 16)

27.5.10 Debug Unit Chip ID Register

Name: DBGU_CIDR

Access Type: Read-only

31	30	29	28	27	26	25	24
EXT	NVPTYP			ARCH			

23	22	21	20	19	18	17	16
ARCH				SRAMSIZ			

15	14	13	12	11	10	9	8
NVPSIZ2				NVPSIZ			

7	6	5	4	3	2	1	0
EPROC			VERSION				

- **VERSION: Version of the Device**
- **EPROC: Embedded Processor**

EPROC			Processor
0	0	1	ARM946ES
0	1	0	ARM7TDMI
1	0	0	ARM920T
1	0	1	ARM926EJ-S

- **NVPSIZ: Nonvolatile Program Memory Size**

NVPSIZ				Size
0	0	0	0	None
0	0	0	1	8K bytes
0	0	1	0	16K bytes
0	0	1	1	32K bytes
0	1	0	0	Reserved
0	1	0	1	64K bytes
0	1	1	0	Reserved
0	1	1	1	128K bytes
1	0	0	0	Reserved
1	0	0	1	256K bytes
1	0	1	0	512K bytes
1	0	1	1	Reserved
1	1	0	0	1024K bytes
1	1	0	1	Reserved
1	1	1	0	2048K bytes
1	1	1	1	Reserved

- **NVPSIZ2 Second Nonvolatile Program Memory Size**

NVPSIZ2				Size
0	0	0	0	None
0	0	0	1	8K bytes
0	0	1	0	16K bytes
0	0	1	1	32K bytes
0	1	0	0	Reserved
0	1	0	1	64K bytes
0	1	1	0	Reserved
0	1	1	1	128K bytes
1	0	0	0	Reserved
1	0	0	1	256K bytes
1	0	1	0	512K bytes
1	0	1	1	Reserved
1	1	0	0	1024K bytes
1	1	0	1	Reserved
1	1	1	0	2048K bytes
1	1	1	1	Reserved

- **SRAMSIZ: Internal SRAM Size**

SRAMSIZ				Size
0	0	0	0	Reserved
0	0	0	1	1K bytes
0	0	1	0	2K bytes
0	0	1	1	Reserved
0	1	0	0	112K bytes
0	1	0	1	4K bytes
0	1	1	0	80K bytes
0	1	1	1	160K bytes
1	0	0	0	8K bytes
1	0	0	1	16K bytes
1	0	1	0	32K bytes
1	0	1	1	64K bytes
1	1	0	0	128K bytes
1	1	0	1	256K bytes
1	1	1	0	96K bytes
1	1	1	1	512K bytes

- **ARCH: Architecture Identifier**

ARCH		
Hex	**Bin**	**Architecture**
0xF0	1111 0001	AT75Cxx Series
0x40	0100 0000	AT91x40 Series
0x63	0110 0011	AT91x63 Series
0x55	0101 0101	AT91x55 Series
0x42	0100 0010	AT91x42 Series
0x92	1001 0010	AT91x92 Series
0x34	0011 0100	AT91x34 Series
0x60	0101 0000	AT91SAM7Axx Series
0x70	0111 0000	AT91SAM7Sxx Series
0x71	0111 0001	AT91SAM7XCxx Series
0x72	0111 0010	AT91SAM7SExx Series
0x73	0111 0011	AT91SAM7Lxx Series
0x75	0111 0101	AT91SAM7Xxx Series
0x19	0001 1001	AT91SAM9xx Series

- **NVPTYP: Nonvolatile Program Memory Type**

NVPTYP			Memory
0	0	0	ROM
0	0	1	ROMless or on-chip Flash
1	0	0	SRAM emulating ROM
0	1	0	Embedded Flash Memory
0	1	1	ROM and Embedded Flash Memory NVPSIZ is ROM size NVPSIZ2 is Flash size

- **EXT: Extension Flag**

0 = Chip ID has a single register definition without extension

1 = An extended Chip ID exists.



27.5.11 Debug Unit Chip ID Extension Register

Name: DBGU_EXID

Access Type: Read-only

31	30	29	28	27	26	25	24
EXID							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
EXID							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
EXID							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EXID							

- **EXID: Chip ID Extension**

Reads 0 if the bit EXT in DBGU_CIDR is 0.

27.5.12 Debug Unit Force NTRST Register

Name: DBGU_FNR

Access T ype: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	FNTRST

- **FNTRST: Force NTRST**

0 = NTRST of the ARM processor’s TAP controller is driven by the ice_nreset signal.

1 = NTRST of the ARM processor’s TAP controller is held low.

28. Parallel Input/Output (PIO) Controller

28.1 Description

The Parallel Input/Output (PIO) Controller manages up to 32 fully programmable input/output lines. Each I/O line may be dedicated as a general-purpose I/O or be assigned to a function of an embedded peripheral. This assures effective optimization of the pins of a product.

Each I/O line is associated with a bit number in all of the 32-bit registers of the 32-bit wide User Interface.

Each I/O line of the PIO Controller features:

- An input change interrupt enabling level change detection on any I/O line.
- A glitch filter providing rejection of pulses lower than one-half of clock cycle.
- Multi-drive capability similar to an open drain I/O line.
- Control of the the pull-up of the I/O line.
- Input visibility and output control.

The PIO Controller also features a synchronous output providing up to 32 bits of data output in a single write operation.

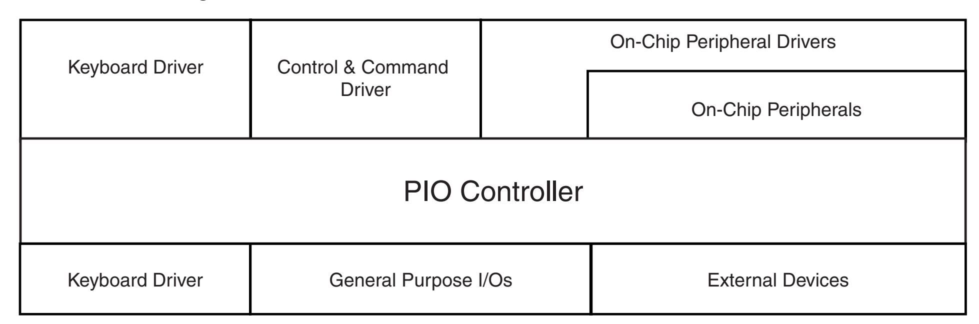


28.2 Block Diagram

Figure 28-1. Block Diagram

PIO Controller
AIC
PIO Interrupt
PMC
PIO Clock
Data, Enable
Embedded Peripheral
Up to 32 peripheral IOs
Data, Enable
Embedded Peripheral
Up to 32 peripheral IOs
PIN 0
PIN 1
PIN 31
Up to 32 pins
APB

Figure 28-2. Application Block Diagram



28.3 Product Dependencies

28.3.1 Pin Multiplexing

Each pin is configurable, according to product definition as either a general-purpose I/O line only, or as an I/O line multiplexed with one or two peripheral I/Os. As the multiplexing is hardware-defined and thus product-dependent, the hardware designer and programmer must carefully determine the configuration of the PIO controllers required by their application. When an I/O line is general-purpose only, i.e., not multiplexed with any peripheral I/O, programming of the PIO Controller regarding the assignment to a peripheral has no effect and only the PIO Controller can control how the pin is driven by the product.

28.3.2 External Interrupt Lines

The interrupt signals FIQ and IRQ0 to IRQn are most generally multiplexed through the PIO Controllers. However, it is not necessary to assign the I/O line to the interrupt function as the PIO Controller has no effect on inputs and the interrupt lines (FIQ or IRQs) are used only as inputs.

28.3.3 Power Management

The Power Management Controller controls the PIO Controller clock in order to save power. Writing any of the registers of the user interface does not require the PIO Controller clock to be enabled. This means that the configuration of the I/O lines does not require the PIO Controller clock to be enabled.

However, when the clock is disabled, not all of the features of the PIO Controller are available. Note that the Input Change Interrupt and the read of the pin level require the clock to be validated.

After a hardware reset, the PIO clock is disabled by default.

The user must configure the Power Management Controller before any access to the input line information.

28.3.4 Interrupt Generation

For interrupt handling, the PIO Controllers are considered as user peripherals. This means that the PIO Controller interrupt lines are connected among the interrupt sources 2 to 31. Refer to the PIO Controller peripheral identifier in the product description to identify the interrupt sources dedicated to the PIO Controllers.

The PIO Controller interrupt can be generated only if the PIO Controller clock is enabled.



28.4 Functional Description

The PIO Controller features up to 32 fully-programmable I/O lines. Most of the control logic associated to each I/O is represented in Figure 28-3. In this description each signal shown represents but one of up to 32 possible indexes.

Figure 28-3. I/O Line Control Logic

PIO_OER[0]
PIO_OSR[0]
PIO_ODR[0]
PIO_PUER[0]
PIO_PUSR[0]
PIO_PUDR[0]
Peripheral A Output Enable
Peripheral B Output Enable
PIO_ASR[0]
PIO_ABSR[0]
PIO_BSR[0]
PIO_PER[0]
PIO_PSR[0]
PIO_PDR[0]
PIO_MDER[0]
PIO_MDSR[0]
PIO_MDDR[0]
Peripheral A Output
Peripheral B Output
PIO_SODR[0]
PIO_ODSR[0]
PIO_CODR[0]
Pad
Peripheral A Input
Peripheral B Input
PIO_PDSR[0]
PIO_ISR[0]
Edge Detector
Glitch Filter
(Up to 32 possible inputs)
PIO Interrupt
PIO_IFER[0]
PIO_IFSR[0]
PIO_IFDR[0]
PIO_IER[0]
PIO_IMR[0]
PIO_IDR[0]
PIO_ISR[31]
PIO_IER[31]
PIO_IMR[31]
PIO_IDR[31]

28.4.1 Pull-up Resistor Control

Each I/O line is designed with an embedded pull-up resistor. The pull-up resistor can be enabled or disabled by writing respectively PIO_PUER (Pull-up Enable Register) and PIO_PUDR (Pull-up Disable Resistor). Writing in these registers results in setting or clearing the corresponding bit in PIO_PUSR (Pull-up Status Register). Reading a 1 in PIO_PUSR means the pull-up is disabled and reading a 0 means the pull-up is enabled.

Control of the pull-up resistor is possible regardless of the configuration of the I/O line.

After reset, all of the pull-ups are enabled, i.e., PIO_PUSR resets at the value 0x0.

28.4.2 I/O Line or Peripheral Function Selection

When a pin is multiplexed with one or two peripheral functions, the selection is controlled with the registers PIO_PER (PIO Enable Register) and PIO_PDR (PIO Disable Register). The register PIO_PSR (PIO Status Register) is the result of the set and clear registers and indicates whether the pin is controlled by the corresponding peripheral or by the PIO Controller. A value of 0 indicates that the pin is controlled by the corresponding on-chip peripheral selected in the PIO_ABSR (AB Select Status Register). A value of 1 indicates the pin is controlled by the PIO controller.

If a pin is used as a general purpose I/O line (not multiplexed with an on-chip peripheral), PIO_PER and PIO_PDR have no effect and PIO_PSR returns 1 for the corresponding bit.

After reset, most generally, the I/O lines are controlled by the PIO controller, i.e., PIO_PSR resets at 1. However, in some events, it is important that PIO lines are controlled by the peripheral (as in the case of memory chip select lines that must be driven inactive after reset or for address lines that must be driven low for booting out of an external memory). Thus, the reset value of PIO_PSR is defined at the product level, depending on the multiplexing of the device.

28.4.3 Peripheral A or B Selection

The PIO Controller provides multiplexing of up to two peripheral functions on a single pin. The selection is performed by writing PIO_ASR (A Select Register) and PIO_BSR (Select B Register). PIO_ABSR (AB Select Status Register) indicates which peripheral line is currently selected. For each pin, the corresponding bit at level 0 means peripheral A is selected whereas the corresponding bit at level 1 indicates that peripheral B is selected.

Note that multiplexing of peripheral lines A and B only affects the output line. The peripheral input lines are always connected to the pin input.

After reset, PIO_ABSR is 0, thus indicating that all the PIO lines are configured on peripheral A. However, peripheral A generally does not drive the pin as the PIO Controller resets in I/O line mode.

Writing in PIO_ASR and PIO_BSR manages PIO_ABSR regardless of the configuration of the pin. However, assignment of a pin to a peripheral function requires a write in the corresponding peripheral selection register (PIO_ASR or PIO_BSR) in addition to a write in PIO_PDR.

28.4.4 Output Control

When the I/0 line is assigned to a peripheral function, i.e., the corresponding bit in PIO_PSR is at 0, the drive of the I/O line is controlled by the peripheral. Peripheral A or B, depending on the value in PIO_ABSR, determines whether the pin is driven or not.



When the I/O line is controlled by the PIO controller, the pin can be configured to be driven. This is done by writing PIO_OER (Output Enable Register) and PIO_ODR (Output Disable Register). The results of these write operations are detected in PIO_OSR (Output Status Register). When a bit in this register is at 0, the corresponding I/O line is used as an input only. When the bit is at 1, the corresponding I/O line is driven by the PIO controller.

The level driven on an I/O line can be determined by writing in PIO_SODR (Set Output Data Register) and PIO_CODR (Clear Output Data Register). These write operations respectively set and clear PIO_ODSR (Output Data Status Register), which represents the data driven on the I/O lines. Writing in PIO_OER and PIO_ODR manages PIO_OSR whether the pin is configured to be controlled by the PIO controller or assigned to a peripheral function. This enables configuration of the I/O line prior to setting it to be managed by the PIO Controller.

Similarly, writing in PIO_SODR and PIO_CODR effects PIO_ODSR. This is important as it defines the first level driven on the I/O line.

28.4.5 Synchronous Data Output

Controlling all parallel busses using several PIOs requires two successive write operations in the PIO_SODR and PIO_CODR registers. This may lead to unexpected transient values. The PIO controller offers a direct control of PIO outputs by single write access to PIO_ODSR (Output Data Status Register). Only bits unmasked by PIO_OSWSR (Output Write Status Register) are written. The mask bits in the PIO_OWSR are set by writing to PIO_OWER (Output Write Enable Register) and cleared by writing to PIO_OWDR (Output Write Disable Register).

After reset, the synchronous data output is disabled on all the I/O lines as PIO_OWSR resets at 0x0.

28.4.6 Multi Drive Control (Open Drain)

Each I/O can be independently programmed in Open Drain by using the Multi Drive feature. This feature permits several drivers to be connected on the I/O line which is driven low only by each device. An external pull-up resistor (or enabling of the internal one) is generally required to guarantee a high level on the line.

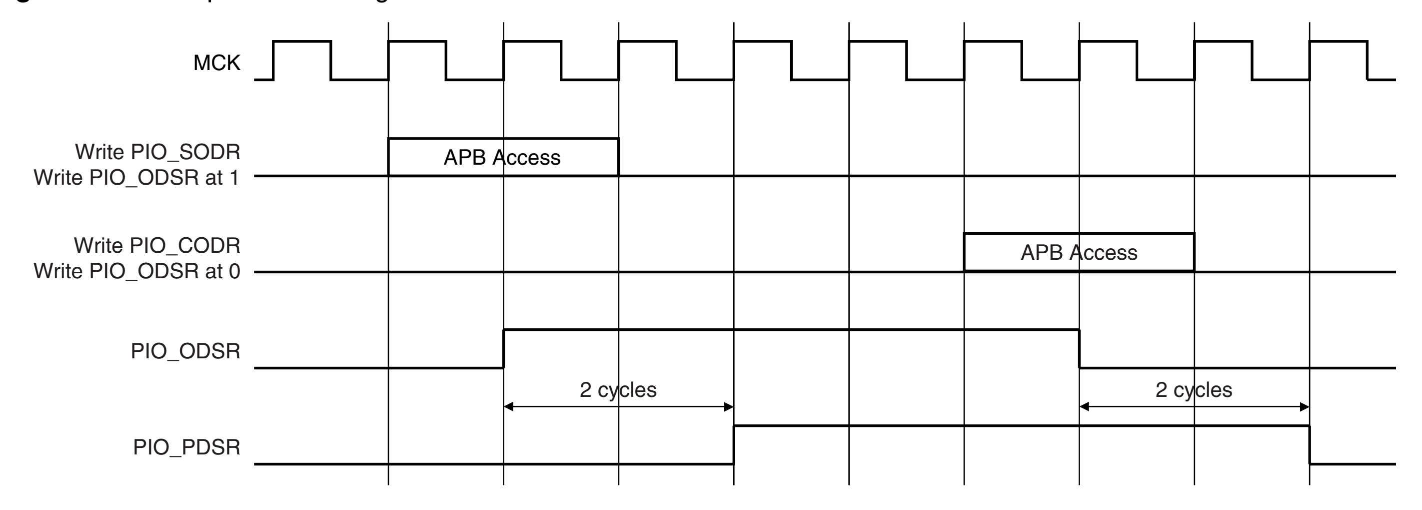
The Multi Drive feature is controlled by PIO_MDER (Multi-driver Enable Register) and PIO_MDDR (Multi-driver Disable Register). The Multi Drive can be selected whether the I/O line is controlled by the PIO controller or assigned to a peripheral function. PIO_MDSR (Multi-driver Status Register) indicates the pins that are configured to support external drivers.

After reset, the Multi Drive feature is disabled on all pins, i.e. PIO_MDSR resets at value 0x0.

28.4.7 Output Line Timings

Figure 28-4 shows how the outputs are driven either by writing PIO_SODR or PIO_CODR, or by directly writing PIO_ODSR. This last case is valid only if the corresponding bit in PIO_OWSR is set. Figure 28-4 also shows when the feedback in PIO_PDSR is available.

Figure 28-4. Output Line Timings



28.4.8 Inputs

The level on each I/O line can be read through PIO_PDSR (Pin Data Status Register). This register indicates the level of the I/O lines regardless of their configuration, whether uniquely as an input or driven by the PIO controller or driven by a peripheral.

Reading the I/O line levels requires the clock of the PIO controller to be enabled, otherwise PIO_PDSR reads the levels present on the I/O line at the time the clock was disabled.

28.4.9 Input Glitch Filtering

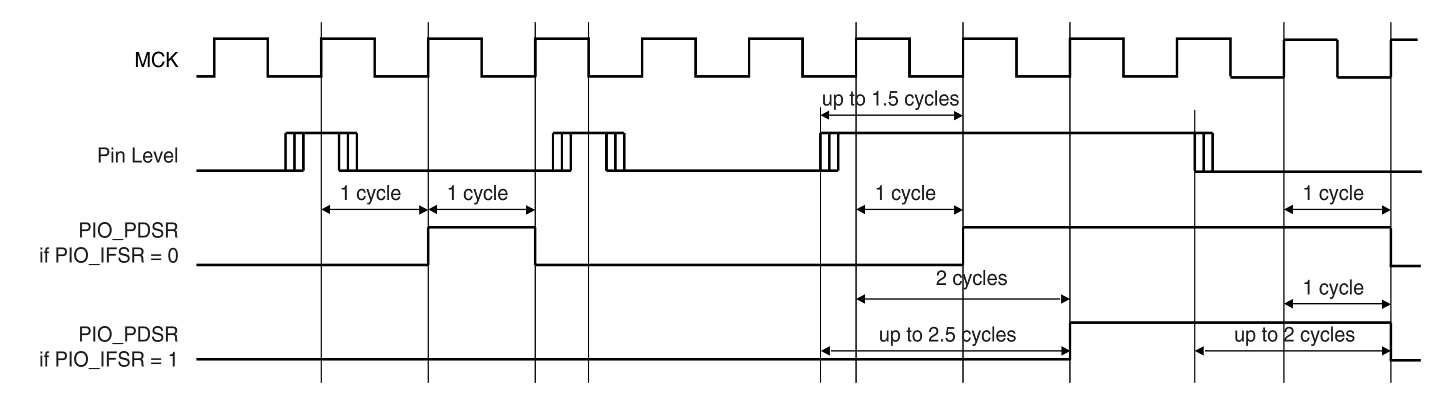
Optional input glitch filters are independently programmable on each I/O line. When the glitch filter is enabled, a glitch with a duration of less than 1/2 Master Clock (MCK) cycle is automatically rejected, while a pulse with a duration of 1 Master Clock cycle or more is accepted. For pulse durations between 1/2 Master Clock cycle and 1 Master Clock cycle the pulse may or may not be taken into account, depending on the precise timing of its occurrence. Thus for a pulse to be visible it must exceed 1 Master Clock cycle, whereas for a glitch to be reliably filtered out, its duration must not exceed 1/2 Master Clock cycle. The filter introduces one Master Clock cycle latency if the pin level change occurs before a rising edge. However, this latency does not appear if the pin level change occurs before a falling edge. This is illustrated in Figure 28-5.

The glitch filters are controlled by the register set; PIO_IFER (Input Filter Enable Register), PIO_IFDR (Input Filter Disable Register) and PIO_IFSR (Input Filter Status Register). Writing PIO_IFER and PIO_IFDR respectively sets and clears bits in PIO_IFSR. This last register enables the glitch filter on the I/O lines.

When the glitch filter is enabled, it does not modify the behavior of the inputs on the peripherals. It acts only on the value read in PIO_PDSR and on the input change interrupt detection. The glitch filters require that the PIO Controller clock is enabled.



Figure 28-5. Input Glitch Filter Timing



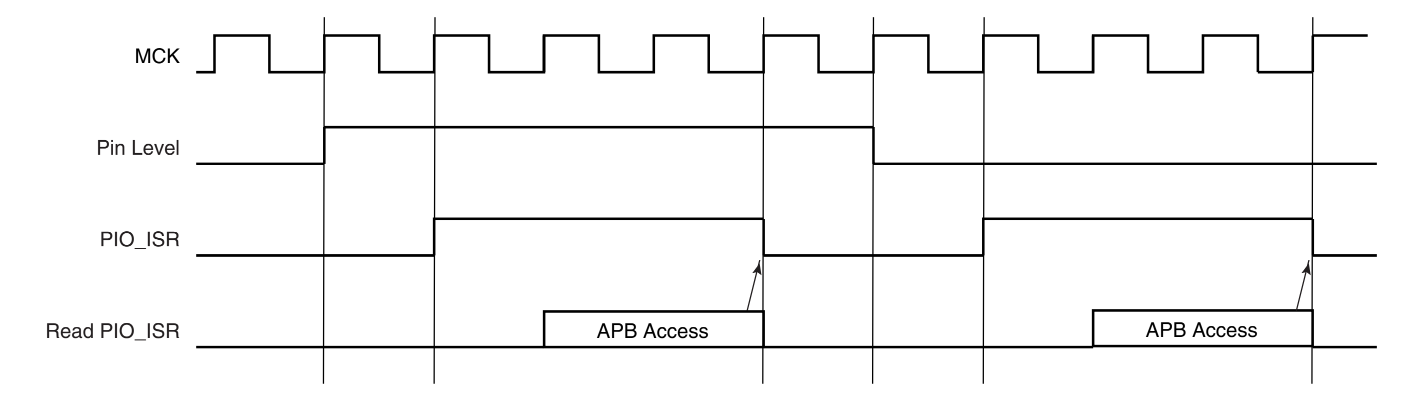
28.4.10 Input Change Interrupt

The PIO Controller can be programmed to generate an interrupt when it detects an input change on an I/O line. The Input Change Interrupt is controlled by writing PIO_IER (Interrupt Enable Register) and PIO_IDR (Interrupt Disable Register), which respectively enable and disable the input change interrupt by setting and clearing the corresponding bit in PIO_IMR (Interrupt Mask Register). As Input change detection is possible only by comparing two successive samplings of the input of the I/O line, the PIO Controller clock must be enabled. The Input Change Interrupt is available, regardless of the configuration of the I/O line, i.e. configured as an input only, controlled by the PIO Controller or assigned to a peripheral function.

When an input change is detected on an I/O line, the corresponding bit in PIO_ISR (Interrupt Status Register) is set. If the corresponding bit in PIO_IMR is set, the PIO Controller interrupt line is asserted. The interrupt signals of the thirty-two channels are ORed-wired together to generate a single interrupt signal to the Advanced Interrupt Controller.

When the software reads PIO_ISR, all the interrupts are automatically cleared. This signifies that all the interrupts that are pending when PIO_ISR is read must be handled.

Figure 28-6. Input Change Interrupt Timings



28.5 I/O Lines Programming Example

The programing example as shown in Table 28-1 below is used to define the following configuration.

- 4-bit output port on I/O lines 0 to 3, (should be written in a single write operation), open-drain, with pull-up resistor
- Four output signals on I/O lines 4 to 7 (to drive LEDs for example), driven high and low, no pull-up resistor
- Four input signals on I/O lines 8 to 11 (to read push-button states for example), with pull-up resistors, glitch filters and input change interrupts
- Four input signals on I/O line 12 to 15 to read an external device status (polled, thus no input change interrupt), no pull-up resistor, no glitch filter
- I/O lines 16 to 19 assigned to peripheral A functions with pull-up resistor
- I/O lines 20 to 23 assigned to peripheral B functions, no pull-up resistor
- I/O line 24 to 27 assigned to peripheral A with Input Change Interrupt and pull-up resistor

Table 28-1. Programming Example

Register	Value to be Written
PIO_PER	0x0000 FFFF
PIO_PDR	0x0FFF 0000
PIO_OER	0x0000 00FF
PIO_ODR	0x0FFF FF00
PIO_IFER	0x0000 0F00
PIO_IFDR	0x0FFF F0FF
PIO_SODR	0x0000 0000
PIO_CODR	0x0FFF FFFF
PIO_IER	0x0F00 0F00
PIO_IDR	0x00FF F0FF
PIO_MDER	0x0000 000F
PIO_MDDR	0x0FFF FFF0
PIO_PUDR	0x00F0 00F0
PIO_PUER	0x0F0F FF0F
PIO_ASR	0x0F0F 0000
PIO_BSR	0x00F0 0000
PIO_OWER	0x0000 000F
PIO_OWDR	0x0FFF FFF0



28.6 Parallel Input/Ouput (PIO) Controller User Interface

Each I/O line controlled by the PIO Controller is associated with a bit in each of the PIO Controller User Interface registers. Each register is 32 bits wide. If a parallel I/O line is not defined, writing to the corresponding bits has no effect. Undefined bits read zero. If the I/O line is not multiplexed with any peripheral, the I/O line is controlled by the PIO Controller and PIO_PSR returns 1 systematically.

Table 28-2. Register Mapping

Offset	Register	Name	Access	Reset Value
0x0000	PIO Enable Register	PIO_PER	Write-only	–
0x0004	PIO Disable Register	PIO_PDR	Write-only	–
0x0008	PIO Status Register (1)	PIO_PSR	Read-only	0x0000 0000
0x000C	Reserved			
0x0010	Output Enable Register	PIO_OER	Write-only	–
0x0014	Output Disable Register	PIO_ODR	Write-only	–
0x0018	Output Status Register	PIO_OSR	Read-only	0x0000 0000
0x001C	Reserved			
0x0020	Glitch Input Filter Enable Register	PIO_IFER	Write-only	–
0x0024	Glitch Input Filter Disable Register	PIO_IFDR	Write-only	–
0x0028	Glitch Input Filter Status Register	PIO_IFSR	Read-only	0x0000 0000
0x002C	Reserved			
0x0030	Set Output Data Register	PIO_SODR	Write-only	–
0x0034	Clear Output Data Register	PIO_CODR	Write-only	–
0x0038	Output Data Status Register(2)	PIO_ODSR	Read-only	0x0000 0000
0x003C	Pin Data Status Register(3)	PIO_PDSR	Read-only	
0x0040	Interrupt Enable Register	PIO_IER	Write-only	–
0x0044	Interrupt Disable Register	PIO_IDR	Write-only	–
0x0048	Interrupt Mask Register	PIO_IMR	Read-only	0x00000000
0x004C	Interrupt Status Register(4)	PIO_ISR	Read-only	0x00000000
0x0050	Multi-driver Enable Register	PIO_MDER	Write-only	–
0x0054	Multi-driver Disable Register	PIO_MDDR	Write-only	–
0x0058	Multi-driver Status Register	PIO_MDSR	Read-only	0x00000000
0x005C	Reserved			
0x0060	Pull-up Disable Register	PIO_PUDR	Write-only	–
0x0064	Pull-up Enable Register	PIO_PUER	Write-only	–
0x0068	Pad Pull-up Status Register	PIO_PUSR	Read-only	0x00000000
0x006C	Reserved			

Table 28-2. Register Mapping (Continued)

Offset	Register	Name	Access	Reset Value
0x0070	Peripheral A Select Register[5]	PIO_ASR	Write-only	–
0x0074	Peripheral B Select Register[5]	PIO_BSR	Write-only	–
0x0078	AB Status Register[5]	PIO_ABSR	Read-only	0x00000000
0x007C to 0x009C	Reserved			
0x00A0	Output Write Enable	PIO_OWER	Write-only	–
0x00A4	Output Write Disable	PIO_OWDR	Write-only	–
0x00A8	Output Write Status Register	PIO_OWSR	Read-only	0x00000000
0x00AC	Reserved			
0x00FC	Version Register	PIO_VERSION	Read-only	–[6]

Notes:
1. Reset value of PIO_PSR depends on the product implementation.
2. PIO_ODSR is Read-only or Read/Write depending on PIO_OWSR I/O lines.
3. Reset value of PIO_PDSR depends on the level of the I/O lines.
4. PIO_ISR is reset at 0x0. However, the first read of the register may read a different value as input changes may have occurred.
5. Only this set of registers clears the status by writing 1 in the first register and sets the status by writing 1 in the second register.
6. The reset value changes with each version of the IP block hardware.



28.6.1 PIO Controller PIO Enable Register

Name: PIO_PER

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: PIO Enable**

0 = No effect.

1 = Enables the PIO to control the corresponding pin (disables peripheral control of the pin).

28.6.2 PIO Controller PIO Disable Register

Name: PIO_PDR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: PIO Disable**

0 = No effect.

1 = Disables the PIO from controlling the corresponding pin (enables peripheral control of the pin).

28.6.3 PIO Controller PIO Status Register

Name: PIO_PSR

Access Type: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: PIO Status**

0 = PIO is inactive on the corresponding I/O line (peripheral is active).

1 = PIO is active on the corresponding I/O line (peripheral is inactive).



28.6.4 PIO Controller Output Enable Register

Name: PIO_OER

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Output Enable**

0 = No effect.

1 = Enables the output on the I/O line.

28.6.5 PIO Controller Output Disable Register

Name: PIO_ODR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Output Disable**

0 = No effect.

1 = Disables the output on the I/O line.

28.6.6 PIO Controller Output Status Register

Name: PIO_OSR

Access Type: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Output Status**

0 = The I/O line is a pure input.

1 = The I/O line is enabled in output.

28.6.7 PIO Controller Input Filter Enable Register

Name: PIO_IFER

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Input Filter Enable**

0 = No effect.

1 = Enables the input glitch filter on the I/O line.

28.6.8 PIO Controller Input Filter Disable Register

Name: PIO_IFDR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Input Filter Disable**

0 = No effect.

1 = Disables the input glitch filter on the I/O line.

28.6.9 PIO Controller Input Filter Status Register

Name: PIO_IFSR

Access Type: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Input Filer Status**

0 = The input glitch filter is disabled on the I/O line.

1 = The input glitch filter is enabled on the I/O line.



28.6.10 PIO Controller Set Output Data Register

Name: PIO_SODR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Set Output Data**

0 = No effect.

1 = Sets the data to be driven on the I/O line.

28.6.11 PIO Controller Clear Output Data Register

Name: PIO_CODR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Set Output Data**

0 = No effect.

1 = Clears the data to be driven on the I/O line.

28.6.12 PIO Controller Output Data Status Register

Name: PIO_ODSR

Access Type: Read-only or Read/Write

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Output Data Status**

0 = The data to be driven on the I/O line is 0.

1 = The data to be driven on the I/O line is 1.

28.6.13 PIO Controller Pin Data Status Register

Name: PIO_PDSR

Access Type: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Output Data Status**

0 = The I/O line is at level 0.

1 = The I/O line is at level 1.

28.6.14 PIO Controller Interrupt Enable Register

Name: PIO_IER

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Input Change Interrupt Enable**

0 = No effect.

1 = Enables the Input Change Interrupt on the I/O line.

28.6.15 PIO Controller Interrupt Disable Register

Name: PIO_IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Input Change Interrupt Disable**

0 = No effect.

1 = Disables the Input Change Interrupt on the I/O line.



28.6.16 PIO Controller Interrupt Mask Register

Name: PIO_IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Input Change Interrupt Mask**

0 = Input Change Interrupt is disabled on the I/O line.

1 = Input Change Interrupt is enabled on the I/O line.

28.6.17 PIO Controller Interrupt Status Register

Name: PIO_ISR

Access Type: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Input Change Interrupt Status**

0 = No Input Change has been detected on the I/O line since PIO_ISR was last read or since reset.

1 = At least one Input Change has been detected on the I/O line since PIO_ISR was last read or since reset.

28.6.18 PIO Multi-driver Enable Register

Name: PIO_MDER

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Multi Drive Enable.**

0 = No effect.

1 = Enables Multi Drive on the I/O line.

28.6.19 PIO Multi-driver Disable Register

Name: PIO_MDDR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Multi Drive Disable.**

0 = No effect.

1 = Disables Multi Drive on the I/O line.



28.6.20 PIO Multi-driver Status Register

Name: PIO_MDSR

Access Type: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Multi Drive Status.**

0 = The Multi Drive is disabled on the I/O line. The pin is driven at high and low level.

1 = The Multi Drive is enabled on the I/O line. The pin is driven at low level only.



28.6.21 PIO Pull Up Disable Register

Name: PIO_PUDR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Pull Up Disable.**

0 = No effect.

1 = Disables the pull up resistor on the I/O line.

28.6.22 PIO Pull Up Enable Register

Name: PIO_PUER

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Pull Up Enable.**

0 = No effect.

1 = Enables the pull up resistor on the I/O line.

28.6.23 PIO Pull Up Status Register

Name: PIO_PUSR

Access Type: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Pull Up Status.**

0 = Pull Up resistor is enabled on the I/O line.

1 = Pull Up resistor is disabled on the I/O line.



28.6.24 PIO Peripheral A Select Register

Name: PIO_ASR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Peripheral A Select.**

0 = No effect.

1 = Assigns the I/O line to the Peripheral A function.

28.6.25 PIO Peripheral B Select Register

Name: PIO_BSR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Peripheral B Select.**

0 = No effect.

1 = Assigns the I/O line to the peripheral B function.



28.6.26 PIO Peripheral A B Status Register

Name: PIO_ABSR

Access Type: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Peripheral A B Status.**

0 = The I/O line is assigned to the Peripheral A.

1 = The I/O line is assigned to the Peripheral B.

28.6.27 PIO Output Write Enable Register

Name: PIO_OWER

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Output Write Enable.**

0 = No effect.

1 = Enables writing PIO_ODSR for the I/O line.

28.6.28 PIO Output Write Disable Register

Name: PIO_OWDR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Output Write Disable.**

0 = No effect.

1 = Disables writing PIO_ODSR for the I/O line.



28.6.29 PIO Output Write Status Register

Name: PIO_OWSR

Access Type: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P23	P22	P21	P20	P19	P18	P17	P16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
P15	P14	P13	P12	P11	P10	P9	P8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Output Write Status.**

0 = Writing PIO_ODSR does not affect the I/O line.

1 = Writing PIO_ODSR affects the I/O line.

28.6.30 PIO Version Register

Name: PIO_VERSION

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	MFN		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	VERSION			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VERSION							

- **VERSION**

Reserved. Value subject to change. No functionality associated. This is the Atmel internal version of the macrocell.

- **MFN**

Reserved. Value subject to change. No functionality associated.

29. Serial Peripheral Interface (SPI)

29.1 Description

The Serial Peripheral Interface (SPI) circuit is a synchronous serial data link that provides communication with external devices in Master or Slave Mode. It also enables communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is essentially a shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the “master”' which controls the data flow, while the other devices act as “slaves'' which have data shifted into and out by the master. Different CPUs can take turn being masters (Multiple Master Protocol opposite to Single Master Protocol where one CPU is always the master while all of the others are always slaves) and one master may simultaneously shift data into multiple slaves. However, only one slave may drive its output to write data back to the master at any given time.

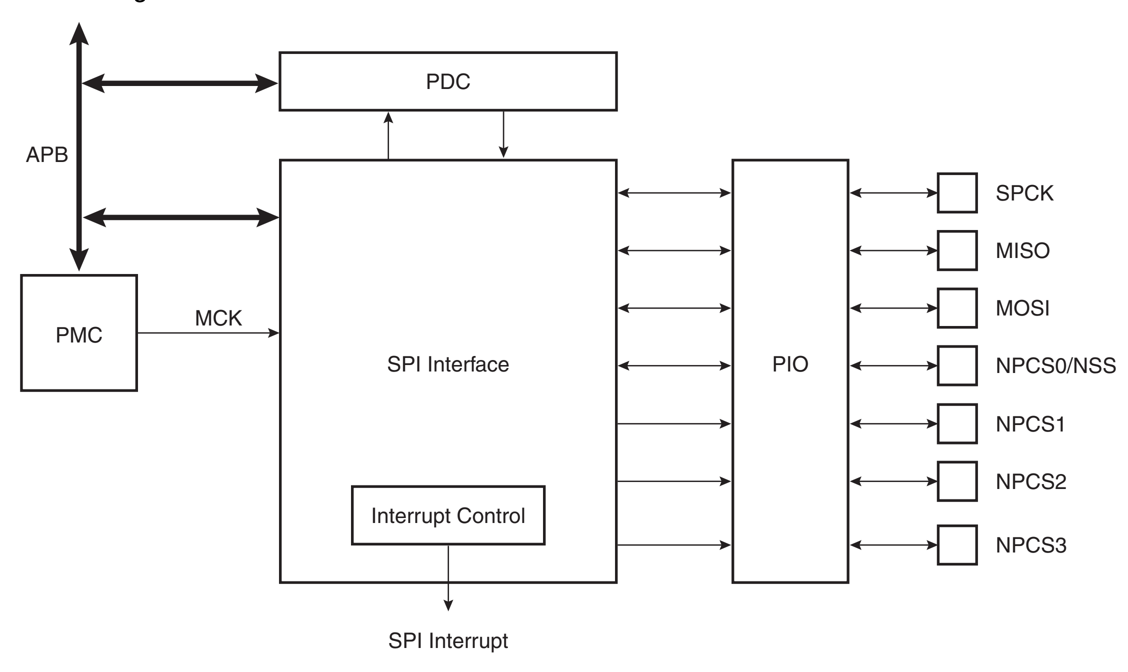
A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

The SPI system consists of two data lines and two control lines:

- Master Out Slave In (MOSI): This data line supplies the output data from the master shifted into the input(s) of the slave(s).
- Master In Slave Out (MISO): This data line supplies the output data from a slave to the input of the master. There may be no more than one slave transmitting data during any particular transfer.
- Serial Clock (SPCK): This control line is driven by the master and regulates the flow of the data bits. The master may transmit data at a variety of baud rates; the SPCK line cycles once for each bit that is transmitted.
- Slave Select (NSS): This control line allows slaves to be turned on and off by hardware.

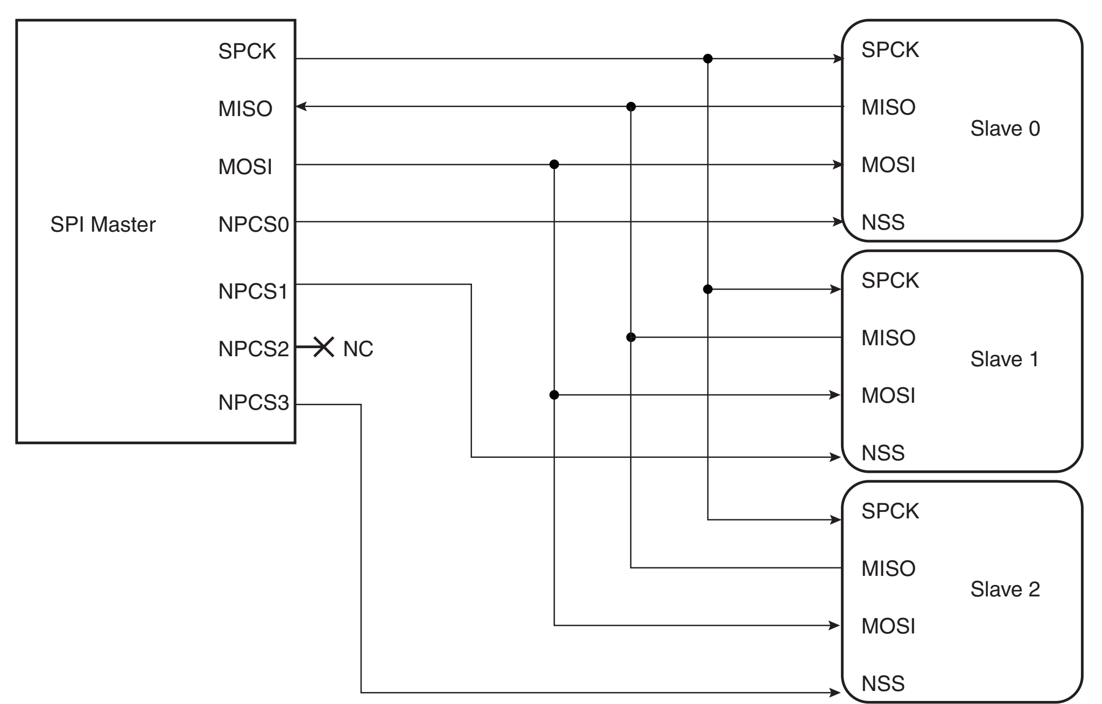
29.2 Block Diagram

Figure 29-1. Block Diagram



29.3 Application Block Diagram

Figure 29-2. Application Block Diagram: Single Master/Multiple Slave Implementation



29.4 Signal Description

Table 29-1. Signal Description

Pin Name	Pin Description	Type	
		Master	Slave
MISO	Master In Slave Out	Input	Output
MOSI	Master Out Slave In	Output	Input
SPCK	Serial Clock	Output	Input
NPCS1-NPCS3	Peripheral Chip Selects	Output	Unused
NPCS0/NSS	Peripheral Chip Select/Slave Select	Output	Input

29.5 Product Dependencies

29.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the SPI pins to their peripheral functions.

29.5.2 Power Management

The SPI may be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the SPI clock.

29.5.3 Interrupt

The SPI interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling the SPI interrupt requires programming the AIC before configuring the SPI.

29.6 Functional Description

29.6.1 Modes of Operation

The SPI operates in Master Mode or in Slave Mode.

Operation in Master Mode is programmed by writing at 1 the MSTR bit in the Mode Register. The pins NPCS0 to NPCS3 are all configured as outputs, the SPCK pin is driven, the MISO line is wired on the receiver input and the MOSI line driven as an output by the transmitter.

If the MSTR bit is written at 0, the SPI operates in Slave Mode. The MISO line is driven by the transmitter output, the MOSI line is wired on the receiver input, the SPCK pin is driven by the transmitter to synchronize the receiver. The NPCS0 pin becomes an input, and is used as a Slave Select signal (NSS). The pins NPCS1 to NPCS3 are not driven and can be used for other purposes.

The data transfers are identically programmable for both modes of operations. The baud rate generator is activated only in Master Mode.

29.6.2 Data Transfer

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the Chip Select Register. The clock phase is programmed with the NCPHA bit. These two parameters determine the edges of the clock signal on which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Thus, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are used and fixed in different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

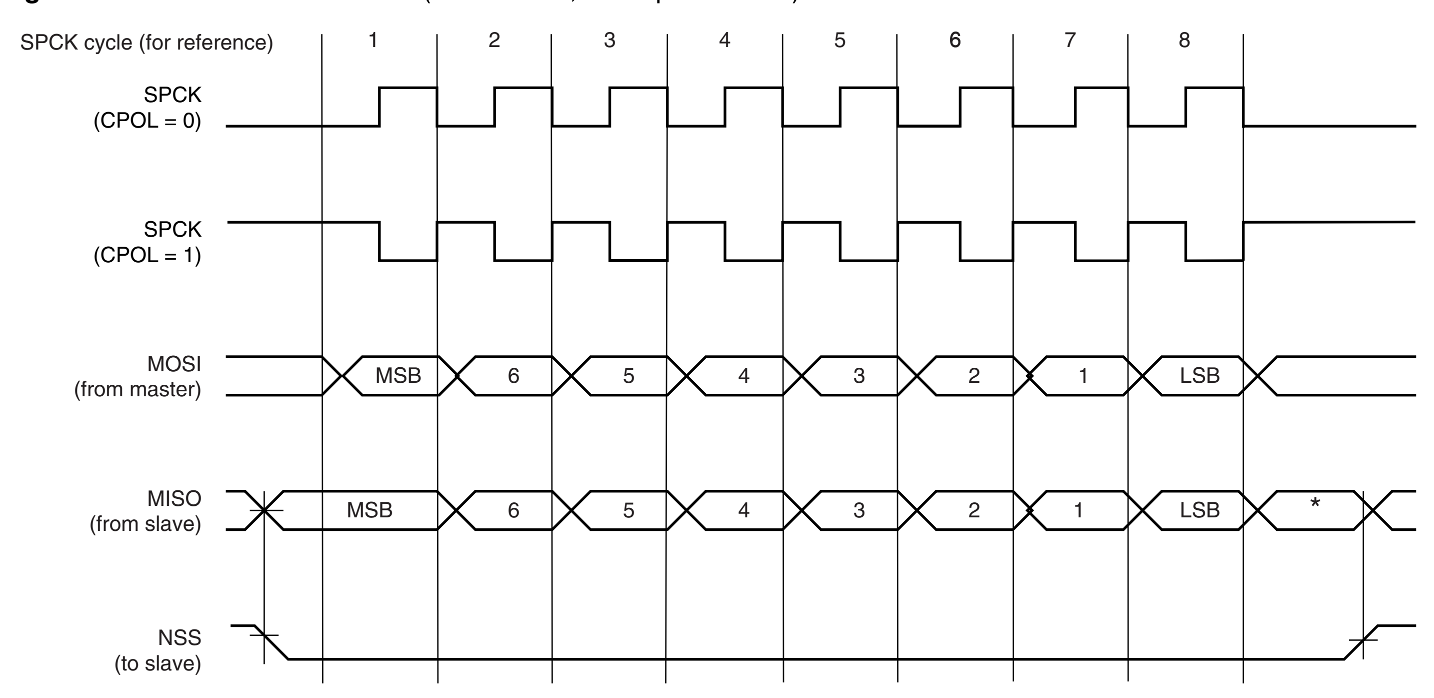
Table 29-2 shows the four modes and corresponding parameter settings.

Table 29-2. SPI Bus Protocol Mode

SPI Mode	CPOL	NCPHA
0	0	1
1	0	0
2	1	1
3	1	0

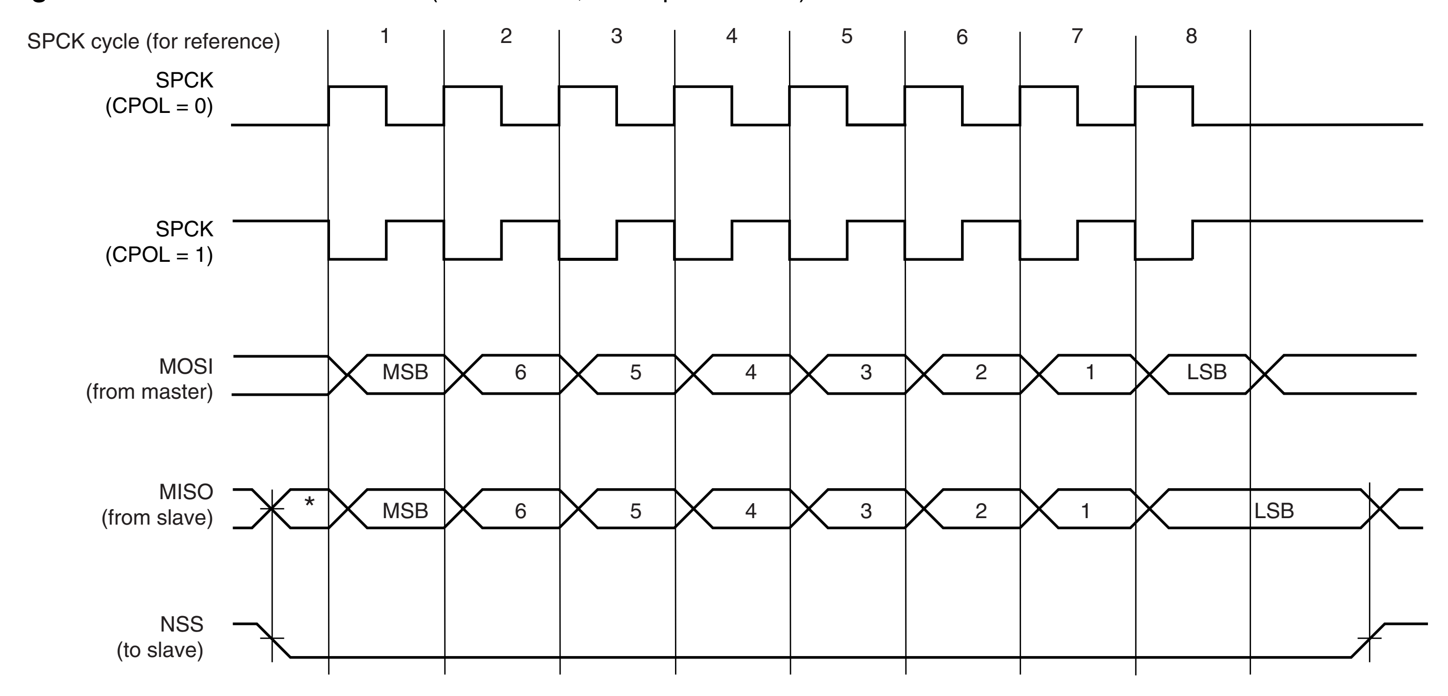
Figure 29-3 and Figure 29-4 show examples of data transfers.

Figure 29-3. SPI Transfer Format (NCPHA = 1, 8 bits per transfer)




* Not defined, but normally MSB of previous character received.

Figure 29-4. SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



* Not defined but normally LSB of previous character transmitted.

29.6.3 Master Mode Operations

When configured in Master Mode, the SPI operates on the clock generated by the internal programmable baud rate generator. It fully controls the data transfers to and from the slave(s) connected to the SPI bus. The SPI drives the chip select line to the slave and the serial clock signal (SPCK).

The SPI features two holding registers, the Transmit Data Register and the Receive Data Register, and a single Shift Register. The holding registers maintain the data flow at a constant rate.

After enabling the SPI, a data transfer begins when the processor writes to the SPI_TDR (Transmit Data Register). The written data is immediately transferred in the Shift Register and transfer on the SPI bus starts. While the data in the Shift Register is shifted on the MOSI line, the MISO line is sampled and shifted in the Shift Register. Transmission cannot occur without reception.

Before writting the TDR, the PCS field must be set in order to select a slave.

If new data is written in SPI_TDR during the transfer, it stays in it until the current transfer is completed. Then, the received data is transferred from the Shift Register to SPI_RDR, the data in SPI_TDR is loaded in the Shift Register and a new transfer starts.

The transfer of a data written in SPI_TDR in the Shift Register is indicated by the TDRE bit (Transmit Data Register Empty) in the Status Register (SPI_SR). When new data is written in SPI_TDR, this bit is cleared. The TDRE bit is used to trigger the Transmit PDC channel.

The end of transfer is indicated by the TXEMPTY flag in the SPI_SR register. If a transfer delay (DLYBCT) is greater than 0 for the last transfer, TXEMPTY is set after the completion of said delay. The master clock (MCK) can be switched off at this time.

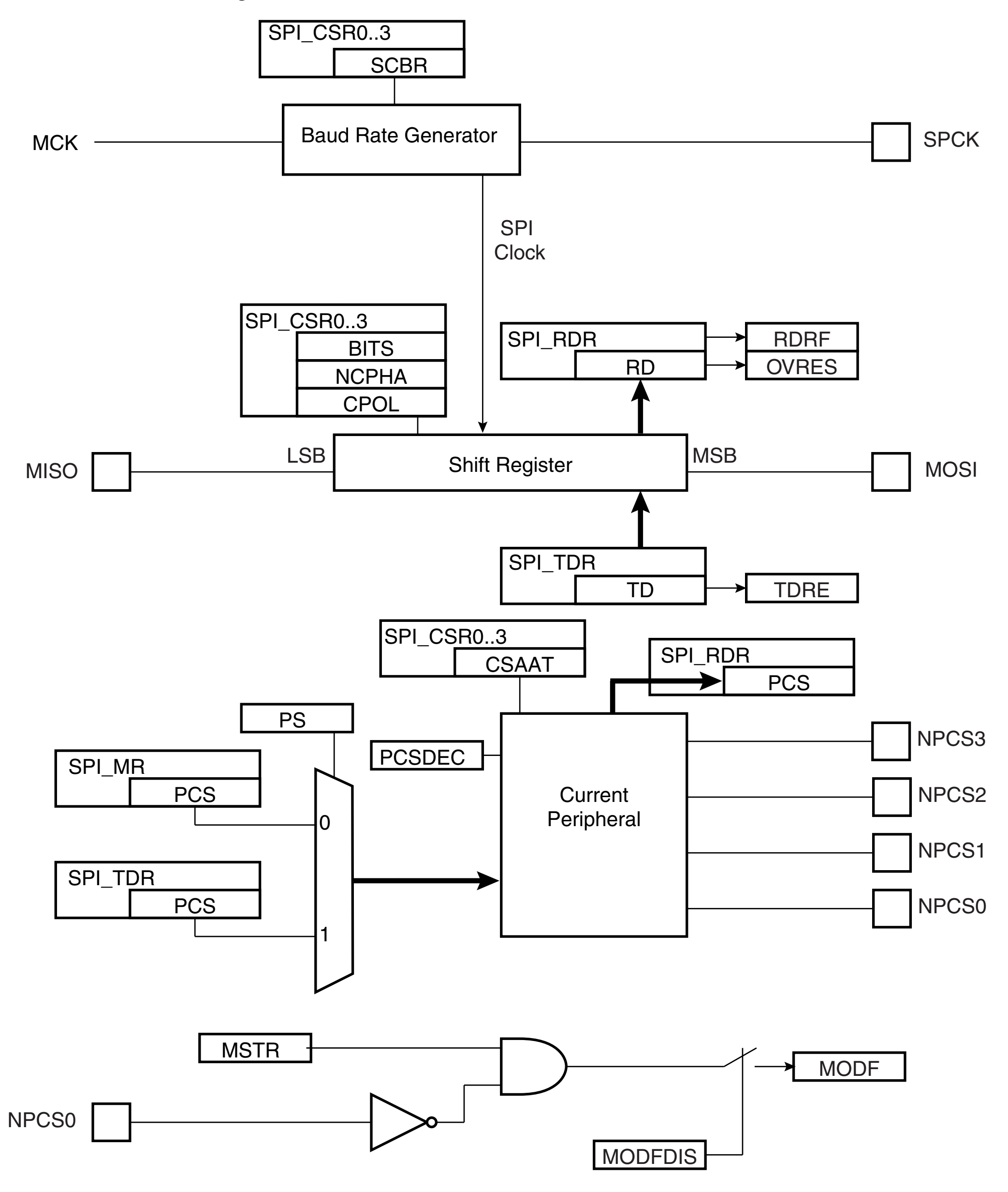
The transfer of received data from the Shift Register in SPI_RDR is indicated by the RDRF bit (Receive Data Register Full) in the Status Register (SPI_SR). When the received data is read, the RDRF bit is cleared.

If the SPI_RDR (Receive Data Register) has not been read before new data is received, the Overrun Error bit (OVRES) in SPI_SR is set. As long as this flag is set, no data is loaded in SPI_RDR. The user has to read the status register to clear the OVRES bit.

Figure 29-6 on page 344 shows a block diagram of the SPI when operating in Master Mode. Figure 29-6 on page 344 shows a flow chart describing how transfers are handled.

29.6.3.1 Master Mode Block Diagram

Figure 29-5. Master Mode Block Diagram



29.6.3.2 Master Mode Flow Diagram

Figure 29-6. Master Mode Flow Diagram S

- NPCS defines the current Chip Select
- CSAAT, DLYBS, DLYBCT refer to the fields of the Chip Select Register corresponding to the Current Chip Select
- When NPCS is 0xF, CSAAT is 0.

SPI Enable
TDRE ?
1
0
CSAAT ?
1
0
PS ?
0
Fixed peripheral
1
Variable peripheral
NPCS = SPI_TDR(PCS)
NPCS = SPI_MR(PCS)
PS ?
0
Fixed peripheral
1
Variable peripheral
SPI_TDR(PCS) = NPCS ?
yes
no
NPCS = 0xF
Delay DLYBCS
NPCS = SPI_TDR(PCS)
SPI_MR(PCS) = NPCS ?
no
NPCS = 0xF
Delay DLYBCS
NPCS = SPI_MR(PCS), SPI_TDR(PCS)
Delay DLYBS
Serializer = SPI_TDR(TD)
TDRE = 1
Data Transfer
SPI_RDR(RD) = Serializer
RDRF = 1
Delay DLYBCT
TDRE ?
0
1
CSAAT ?
1
0
NPCS = 0xF
Delay DLYBCS

29.6.3.3 *Clock Generation*

The SPI Baud rate clock is generated by dividing the Master Clock (MCK) by a value between 1 and 255.

This allows a maximum operating baud rate at up to Master Clock and a minimum operating baud rate of MCK divided by 255.

Programming the SCBR field at 0 is forbidden. Triggering a transfer while SCBR is at 0 can lead to unpredictable results.

At reset, SCBR is 0 and the user has to program it at a valid value before performing the first transfer.

The divisor can be defined independently for each chip select, as it has to be programmed in the SCBR field of the Chip Select Registers. This allows the SPI to automatically adapt the baud rate for each interfaced peripheral without reprogramming.

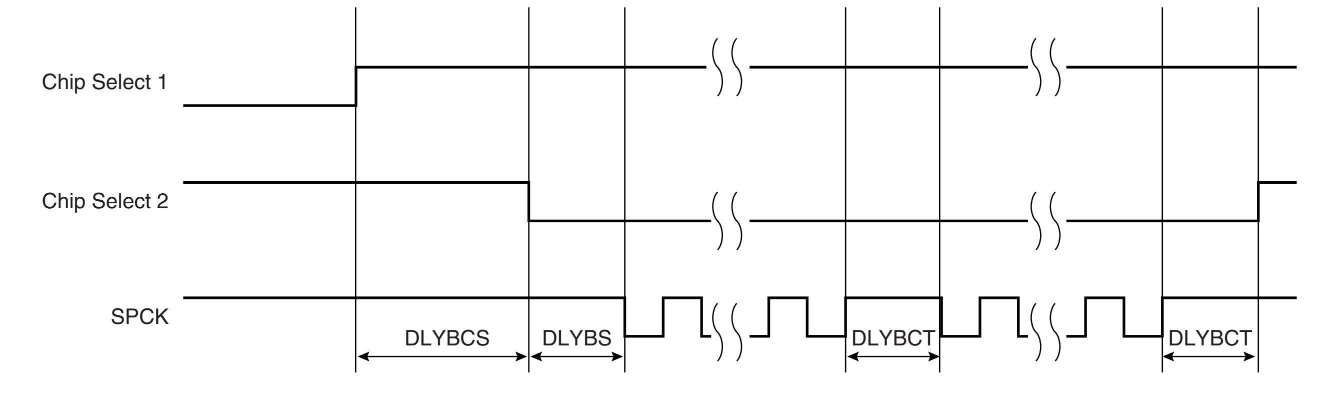
29.6.3.4 *Transfer Delays*

Figure 29-7 shows a chip select transfer change and consecutive transfers on the same chip select. Three delays can be programmed to modify the transfer waveforms:

- The delay between chip selects, programmable only once for all the chip selects by writing the DLYBCS field in the Mode Register. Allows insertion of a delay between release of one chip select and before assertion of a new one.
- The delay before SPCK, independently programmable for each chip select by writing the field DLYBS. Allows the start of SPCK to be delayed after the chip select has been asserted.
- The delay between consecutive transfers, independently programmable for each chip select by writing the DLYBCT field. Allows insertion of a delay between two transfers occurring on the same chip select

These delays allow the SPI to be adapted to the interfaced peripherals and their speed and bus release time.

Figure 29-7. Programmable Delays



29.6.3.5 Peripheral Selection

The serial peripherals are selected through the assertion of the NPCS0 to NPCS3 signals. By default, all the NPCS signals are high before and after each transfer.

The peripheral selection can be performed in two different ways:

- Fixed Peripheral Select: SPI exchanges data with only one peripheral
- Variable Peripheral Select: Data can be exchanged with more than one peripheral

Fixed Peripheral Select is activated by writing the PS bit to zero in SPI_MR (Mode Register). In this case, the current peripheral is defined by the PCS field in SPI_MR and the PCS field in the SPI_TDR has no effect.

Variable Peripheral Select is activated by setting PS bit to one. The PCS field in SPI_TDR is used to select the current peripheral. This means that the peripheral selection can be defined for each new data.

The Fixed Peripheral Selection allows buffer transfers with a single peripheral. Using the PDC is an optimal means, as the size of the data transfer between the memory and the SPI is either 8 bits or 16 bits. However, changing the peripheral selection requires the Mode Register to be reprogrammed.

The Variable Peripheral Selection allows buffer transfers with multiple peripherals without reprogramming the Mode Register. Data written in SPI_TDR is 32 bits wide and defines the real data to be transmitted and the peripheral it is destined to. Using the PDC in this mode requires 32-bit wide buffers, with the data in the LSBs and the PCS and LASTXFER fields in the MSBs, however the SPI still controls the number of bits (8 to16) to be transferred through MISO and MOSI lines with the chip select configuration registers. This is not the optimal means in term of memory size for the buffers, but it provides a very effective means to exchange data with several peripherals without any intervention of the processor.

29.6.3.6 Peripheral Chip Select Decoding

The user can program the SPI to operate with up to 15 peripherals by decoding the four Chip Select lines, NPCS0 to NPCS3 with an external logic. This can be enabled by writing the PCS-DEC bit at 1 in the Mode Register (SPI_MR).

When operating without decoding, the SPI makes sure that in any case only one chip select line is activated, i.e. driven low at a time. If two bits are defined low in a PCS field, only the lowest numbered chip select is driven low.

When operating with decoding, the SPI directly outputs the value defined by the PCS field of either the Mode Register or the Transmit Data Register (depending on PS).

As the SPI sets a default value of 0xF on the chip select lines (i.e. all chip select lines at 1) when not processing any transfer, only 15 peripherals can be decoded.

The SPI has only four Chip Select Registers, not 15. As a result, when decoding is activated, each chip select defines the characteristics of up to four peripherals. As an example, SPI_CRS0 defines the characteristics of the externally decoded peripherals 0 to 3, corresponding to the PCS values 0x0 to 0x3. Thus, the user has to make sure to connect compatible peripherals on the decoded chip select lines 0 to 3, 4 to 7, 8 to 11 and 12 to 14.

29.6.3.7 *Peripheral Deselection*

When operating normally, as soon as the transfer of the last data written in SPI_TDR is completed, the NPCS lines all rise. This might lead to runtime error if the processor is too long in responding to an interrupt, and thus might lead to difficulties for interfacing with some serial peripherals requiring the chip select line to remain active during a full set of transfers.

To facilitate interfacing with such devices, the Chip Select Register can be programmed with the CSAAT bit (Chip Select Active After Transfer) at 1. This allows the chip select lines to remain in their current state (low = active) until transfer to another peripheral is required.

Figure 29-8 shows different peripheral deselection cases and the effect of the CSAAT bit.

Figure 29-8. Peripheral Deselection

CSAAT = 0

CSAAT = 1

29.6.3.8 *Mode Fault Detection*

A mode fault is detected when the SPI is programmed in Master Mode and a low level is driven by an external master on the NPCS0/NSS signal. As this pin is generally configured in open-drain, it is important that a pull up resistor is connected on the NPCS0 line, so that a high level is guaranteed and no spurious mode fault is detected.

When a mode fault is detected, the MODF bit in the SPI_SR is set until the SPI_SR is read and the SPI is automatically disabled until re-enabled by writing the SPIEN bit in the SPI_CR (Control Register) at 1.

By default, the Mode Fault detection circuitry is enabled. The user can disable Mode Fault detection by setting the MODFDIS bit in the SPI Mode Register (SPI_MR).

29.6.4 SPI Slave Mode

When operating in Slave Mode, the SPI processes data bits on the clock provided on the SPI clock pin (SPCK).

The SPI waits for NSS to go active before receiving the serial clock from an external master. When NSS falls, the clock is validated on the serializer, which processes the number of bits defined by the BITS field of the Chip Select Register 0 (SPI_CSR0). These bits are processed following a phase and a polarity defined respectively by the NCPHA and CPOL bits of the SPI_CSR0. Note that BITS, CPOL and NCPHA of the other Chip Select Registers have no effect when the SPI is programmed in Slave Mode.

The bits are shifted out on the MISO line and sampled on the MOSI line.

When all the bits are processed, the received data is transferred in the Receive Data Register and the RDRF bit rises. If RDRF is already high when the data is transferred, the Overrun bit rises and the data transfer to SPI_RDR is aborted.

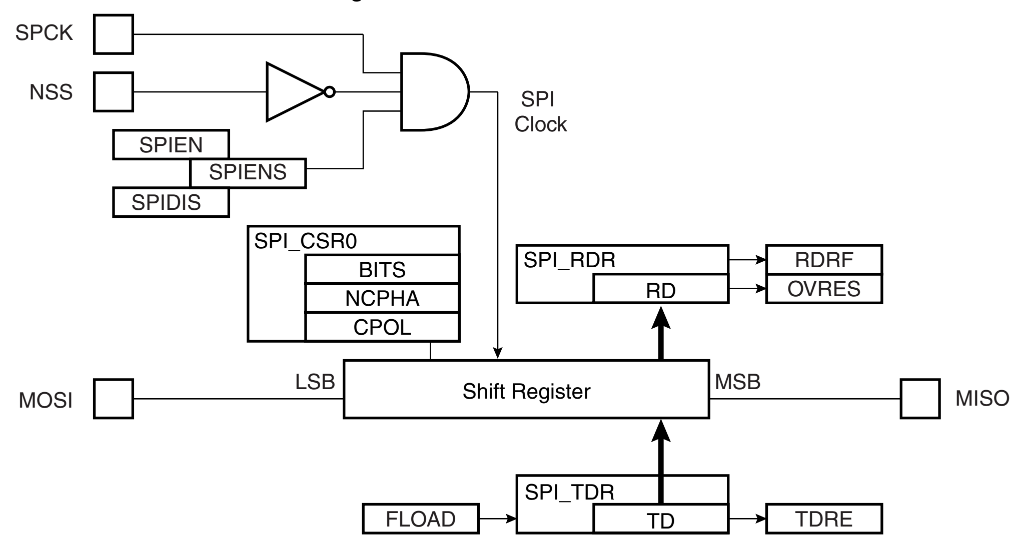
When a transfer starts, the data shifted out is the data present in the Shift Register. If no data has been written in the Transmit Data Register (SPI_TDR), the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift Register resets at 0.

When a first data is written in SPI_TDR, it is transferred immediately in the Shift Register and the TDRE bit rises. If new data is written, it remains in SPI_TDR until a transfer occurs, i.e. NSS falls and there is a valid clock on the SPCK pin. When the transfer occurs, the last data written in SPI_TDR is transferred in the Shift Register and the TDRE bit rises. This enables frequent updates of critical variables with single transfers.

Then, a new data is loaded in the Shift Register from the Transmit Data Register. In case no character is ready to be transmitted, i.e. no character has been written in SPI_TDR since the last load from SPI_TDR to the Shift Register, the Shift Register is not modified and the last received character is retransmitted.

Figure 29-9 shows a block diagram of the SPI when operating in Slave Mode.

Figure 29-9. Slave Mode Functional Block Diagram



29.7 Serial Peripheral Interface (SPI) User Interface

Table 29-3. SPI Register Mapping

Offset	Register	Register Name	Access	Reset
0x00	Control Register	SPI_CR	Write-only	---
0x04	Mode Register	SPI_MR	Read/Write	0x0
0x08	Receive Data Register	SPI_RDR	Read-only	0x0
0x0C	Transmit Data Register	SPI_TDR	Write-only	---
0x10	Status Register	SPI_SR	Read-only	0x000000F0
0x14	Interrupt Enable Register	SPI_IER	Write-only	---
0x18	Interrupt Disable Register	SPI_IDR	Write-only	---
0x1C	Interrupt Mask Register	SPI_IMR	Read-only	0x0
0x20 - 0x2C	Reserved			
0x30	Chip Select Register 0	SPI_CSR0	Read/Write	0x0
0x34	Chip Select Register 1	SPI_CSR1	Read/Write	0x0
0x38	Chip Select Register 2	SPI_CSR2	Read/Write	0x0
0x3C	Chip Select Register 3	SPI_CSR3	Read/Write	0x0
0x004C - 0x00F8	Reserved	–	–	–
0x004C - 0x00FC	Reserved	–	–	–
0x100 - 0x124	Reserved for the PDC			

29.7.1 SPI Control Register

Name: SPI_CR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	LASTXFER
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SWRST	–	–	–	–	–	SPIDIS	SPIEN

- **SPIEN: SPI Enable**

0 = No effect.

1 = Enables the SPI to transfer and receive data.

- **SPIDIS: SPI Disable**

0 = No effect.

1 = Disables the SPI.

As soon as SPIDIS is set, SPI finishes its tranfer.

All pins are set in input mode and no data is received or transmitted.

If a transfer is in progress, the transfer is finished before the SPI is disabled.

If both SPIEN and SPIDIS are equal to one when the control register is written, the SPI is disabled.

- **SWRST: SPI Software Reset**

0 = No effect.

1 = Reset the SPI. A software-triggered hardware reset of the SPI interface is performed.

- **LASTXFER: Last Transfer**

0 = No effect.

1 = The current NPCS will be deasserted after the character written in TD has been transferred. When CSAAT is set, this allows to close the communication with the current serial peripheral by raising the corresponding NPCS line as soon as TD transfer has completed.

29.7.2 SPI Mode Register

Name: SPI_MR

Access Type: Read/Write

31	30	29	28	27	26	25	24
DLYBCS							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	PCS			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LLB	–	–	MODFDIS		PCSDEC	PS	MSTR

- **MSTR: Master/Slave Mode**

0 = SPI is in Slave mode.

1 = SPI is in Master mode.

- **PS: Peripheral Select**

0 = Fixed Peripheral Select.

1 = Variable Peripheral Select.

- **PCSDEC: Chip Select Decode**

0 = The chip selects are directly connected to a peripheral device.

1 = The four chip select lines are connected to a 4- to 16-bit decoder.

When PCSDEC equals one, up to 15 Chip Select signals can be generated with the four lines using an external 4- to 16-bit decoder. The Chip Select Registers define the characteristics of the 15 chip selects according to the following rules:

SPI_CSR0 defines peripheral chip select signals 0 to 3.

SPI_CSR1 defines peripheral chip select signals 4 to 7.

SPI_CSR2 defines peripheral chip select signals 8 to 11.

SPI_CSR3 defines peripheral chip select signals 12 to 14.

- **MODFDIS: Mode Fault Detection**

0 = Mode fault detection is enabled.

1 = Mode fault detection is disabled.

- **LLB: Local Loopback Enable**

0 = Local loopback path disabled.

1 = Local loopback path enabled (

LLB controls the local loopback on the data serializer for testing in Master Mode only. (MISO is internally connected on MOSI.)

- **PCS: Peripheral Chip Select**

This field is only used if Fixed Peripheral Select is active (PS = 0).

If PCSDEC = 0:

PCS = xxx0 NPCS[3:0] = 1110

PCS = xx01 NPCS[3:0] = 1101

PCS = x011 NPCS[3:0] = 1011

PCS = 0111 NPCS[3:0] = 0111

PCS = 1111 forbidden (no peripheral is selected)

(x = don't care)

If PCSDEC = 1:

NPCS[3:0] output signals = PCS.

- **DLYBCS: Delay Between Chip Selects**

This field defines the delay from NPCS inactive to the activation of another NPCS. The DLYBCS time guarantees non-overlapping chip selects and solves bus contentions in case of peripherals having long data float times.

If DLYBCS is less than or equal to six, six MCK periods will be inserted by default.

Otherwise, the following equation determines the delay:

$$\text{Delay Between Chip Selects} = \frac{DLYBCS}{MCK}$$

29.7.3 SPI Receive Data Register

Name: SPI_RDR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	PCS			

15	14	13	12	11	10	9	8
RD							

7	6	5	4	3	2	1	0
RD							

- **RD: Receive Data**

Data received by the SPI Interface is stored in this register right-justified. Unused bits read zero.

- **PCS: Peripheral Chip Select**

In Master Mode only, these bits indicate the value on the NPCS pins at the end of a transfer. Otherwise, these bits read zero.

29.7.4 SPI Transmit Data Register

Name: SPI_TDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	LASTXFER
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	PCS			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TD							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TD							

- **TD: Transmit Data**

Data to be transmitted by the SPI Interface is stored in this register. Information to be transmitted must be written to the transmit data register in a right-justified format.

PCS: Peripheral Chip Select

This field is only used if Variable Peripheral Select is active (PS = 1).

If PCSDEC = 0:

PCS = xxx0 NPCS[3:0] = 1110

PCS = xx01 NPCS[3:0] = 1101

PCS = x011 NPCS[3:0] = 1011

PCS = 0111 NPCS[3:0] = 0111

PCS = 1111 forbidden (no peripheral is selected)

(x = don't care)

If PCSDEC = 1:

NPCS[3:0] output signals = PCS

- **LASTXFER: Last Transfer**

0 = No effect.

1 = The current NPCS will be deasserted after the character written in TD has been transferred. When CSAAT is set, this allows to close the communication with the current serial peripheral by raising the corresponding NPCS line as soon as TD transfer has completed.

This field is only used if Variable Peripheral Select is active (PS = 1).

29.7.5 SPI Status Register

Name: SPI_SR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	SPIENS
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	TXEMPTY	NSSR
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full**

0 = No data has been received since the last read of SPI_RDR

1 = Data has been received and the received data has been transferred from the serializer to SPI_RDR since the last read of SPI_RDR.

- **TDRE: Transmit Data Register Empty**

0 = Data has been written to SPI_TDR and not yet transferred to the serializer.

1 = The last data written in the Transmit Data Register has been transferred to the serializer.

TDRE equals zero when the SPI is disabled or at reset. The SPI enable command sets this bit to one.

- **MODF: Mode Fault Error**

0 = No Mode Fault has been detected since the last read of SPI_SR.

1 = A Mode Fault occurred since the last read of the SPI_SR.

- **OVRES: Overrun Error Status**

0 = No overrun has been detected since the last read of SPI_SR.

1 = An overrun has occurred since the last read of SPI_SR.

An overrun occurs when SPI_RDR is loaded at least twice from the serializer since the last read of the SPI_RDR.

- **ENDRX: End of RX buffer**

0 = The Receive Counter Register has not reached 0 since the last write in SPI_RCR[(1)] or SPI_RNCR[(1)].

1 = The Receive Counter Register has reached 0 since the last write in SPI_RCR[(1)] or SPI_RNCR[(1)].

- **ENDTX: End of TX buffer**

0 = The Transmit Counter Register has not reached 0 since the last write in SPI_TCR[(1)] or SPI_TNCR[(1)].

1 = The Transmit Counter Register has reached 0 since the last write in SPI_TCR[(1)] or SPI_TNCR[(1)].

- **RXBUFF: RX Buffer Full**

0 = SPI_RCR[(1)] or SPI_RNCR[(1)] has a value other than 0.

1 = Both SPI_RCR[(1)] and SPI_RNCR[(1)] have a value of 0.

- **TXBUFE: TX Buffer Empty**

0 = SPI_TCR[(1)] or SPI_TNCR[(1)] has a value other than 0.

1 = Both SPI_TCR[(1)] and SPI_TNCR[(1)] have a value of 0.

- **NSSR: NSS Rising**

0 = No rising edge detected on NSS pin since last read.

1 = A rising edge occurred on NSS pin since last read.

- **TXEMPTY: Transmission Registers Empty**

0 = As soon as data is written in SPI_TDR.

1 = SPI_TDR and internal shifter are empty. If a transfer delay has been defined, TXEMPTY is set after the completion of such delay.

- **SPIENS: SPI Enable Status**

0 = SPI is disabled.

1 = SPI is enabled.

Note: 1. SPI_RCR, SPI_RNCR, SPI_TCR, SPI_TNCR are physically located in the PDC.



29.7.6 SPI Interrupt Enable Register

Name: SPI_IER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	TXEMPTY	NSSR
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full Interrupt Enable**
- **TDRE: SPI Transmit Data Register Empty Interrupt Enable**
- **MODF: Mode Fault Error Interrupt Enable**
- **OVRES: Overrun Error Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable**
- **ENDTX: End of Transmit Buffer Interrupt Enable**
- **RXBUFF: Receive Buffer Full Interrupt Enable**
- **TXBUFE: Transmit Buffer Empty Interrupt Enable**
- **TXEMPTY: Transmission Registers Empty Enable**
- **NSSR: NSS Rising Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

29.7.7 SPI Interrupt Disable Register

Name: SPI_IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR

7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full Interrupt Disable**
- **TDRE: SPI Transmit Data Register Empty Interrupt Disable**
- **MODF: Mode Fault Error Interrupt Disable**
- **OVRES: Overrun Error Interrupt Disable**
- **ENDRX: End of Receive Buffer Interrupt Disable**
- **ENDTX: End of Transmit Buffer Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**
- **TXBUFE: Transmit Buffer Empty Interrupt Disable**
- **TXEMPTY: Transmission Registers Empty Disable**
- **NSSR: NSS Rising Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.



29.7.8 SPI Interrupt Mask Register

Name: SPI_IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	TXEMPTY	NSSR
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full Interrupt Mask**
- **TDRE: SPI Transmit Data Register Empty Interrupt Mask**
- **MODF: Mode Fault Error Interrupt Mask**
- **OVRES: Overrun Error Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **ENDTX: End of Transmit Buffer Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**
- **TXBUFE: Transmit Buffer Empty Interrupt Mask**
- **TXEMPTY: Transmission Registers Empty Mask**
- **NSSR: NSS Rising Interrupt Mask**

0 = The corresponding interrupt is not enabled.

1 = The corresponding interrupt is enabled.

29.7.9 SPI Chip Select Register

Name: SPI_CSR0... SPI_CSR3

Access Type: Read/Write

31	30	29	28	27	26	25	24
DLYBCT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DLYBS							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SCBR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BITS				CSAAT	–	NCPHA	CPOL

- **CPOL: Clock Polarity**

0 = The inactive state value of SPCK is logic level zero.

1 = The inactive state value of SPCK is logic level one.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with NCPHA to produce the required clock/data relationship between master and slave devices.

- **NCPHA: Clock Phase**

0 = Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.

1 = Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.

NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. NCPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.

- **CSAAT: Chip Select Active After Transfer**

0 = The Peripheral Chip Select Line rises as soon as the last transfer is achieved.

1 = The Peripheral Chip Select does not rise after the last transfer is achieved. It remains active until a new transfer is requested on a different chip select.

- **BITS: Bits Per Transfer**

The BITS field determines the number of data bits transferred. Reserved values should not be used.

BITS	Bits Per Transfer
0000	8
0001	9
0010	10
0011	11
0100	12
0101	13
0110	14
0111	15
1000	16
1001	Reserved
1010	Reserved
1011	Reserved

BITS	Bits Per Transfer
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

- **SCBR: Serial Clock Baud Rate**

In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the Master Clock MCK. The Baud rate is selected by writing a value from 1 to 255 in the SCBR field. The following equations determine the SPCK baud rate:

$$\text{SPCK Baudrate} = \frac{MCK}{SCBR}$$

Programming the SCBR field at 0 is forbidden. Triggering a transfer while SCBR is at 0 can lead to unpredictable results.

At reset, SCBR is 0 and the user has to program it at a valid value before performing the first transfer.

- **DLYBS: Delay Before SPCK**

This field defines the delay from NPCS valid to the first valid SPCK transition.

When DLYBS equals zero, the NPCS valid to SPCK transition is 1/2 the SPCK clock period.

Otherwise, the following equations determine the delay:

$$\text{Delay Before SPCK} = \frac{DLYBS}{MCK}$$

- **DLYBCT: Delay Between Consecutive Transfers**

This field defines the delay between two consecutive transfers with the same peripheral without removing the chip select. The delay is always inserted after each transfer and before removing the chip select if needed.

When DLYBCT equals zero, no delay between consecutive transfers is inserted and the clock keeps its duty cycle over the character transfers.

Otherwise, the following equation determines the delay:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{MCK} + \frac{SCBR}{2MCK}$$

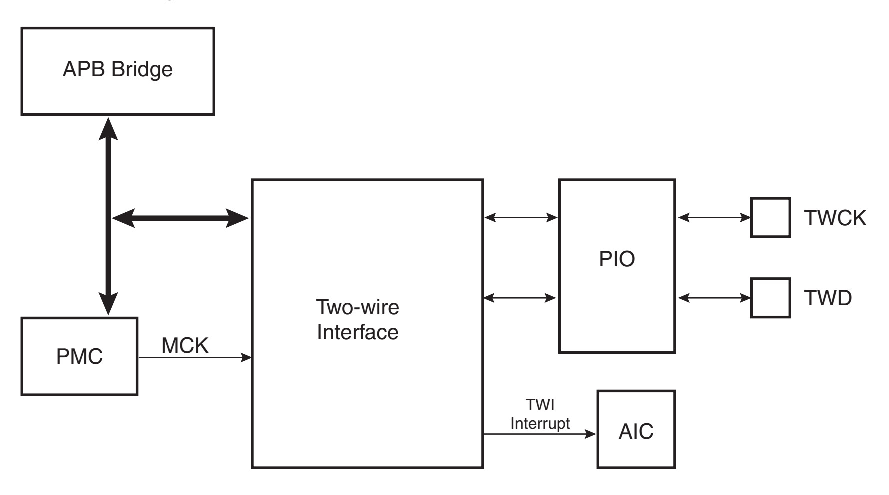
30. Two-wire Interface (TWI)

30.1 Overview

The Two-wire Interface (TWI) interconnects components on a unique two-wire bus, made up of one clock line and one data line with speeds of up to 400 Kbits per second, based on a byte-oriented transfer format. It can be used with any Atmel two-wire bus Serial EEPROM. The TWI is programmable as a master with sequential or single-byte access. A configurable baud rate generator permits the output data rate to be adapted to a wide range of core clock frequencies.

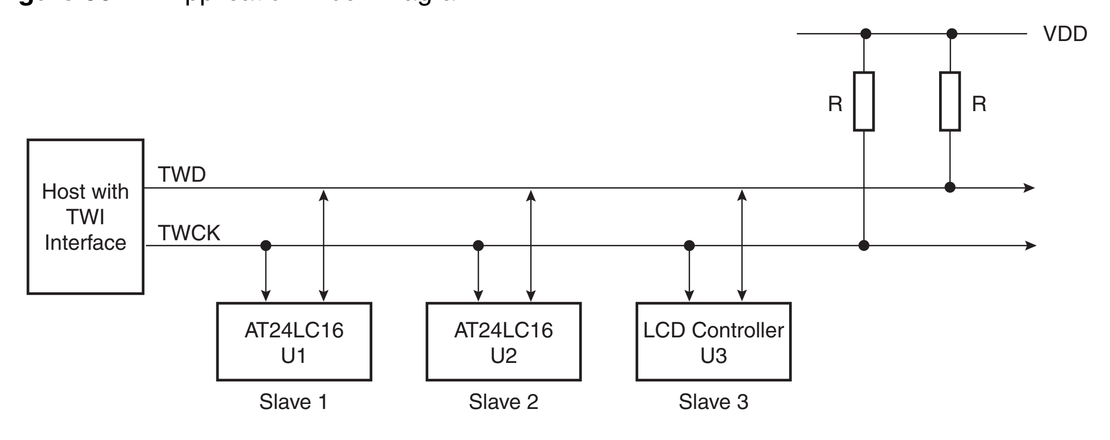
30.2 Block Diagram

Figure 30-1. Block Diagram



30.3 Application Block Diagram

Figure 30-2. Application Block Diagram



30.4 Product Dependencies

30.4.1 I/O Lines Description

Table 30-1. I/O Lines Description

Pin Name	Pin Description	Type
TWD	Two-wire Serial Data	Input/Output
TWCK	Two-wire Serial Clock	Input/Output

Both TWD and TWCK are bidirectional lines, connected to a positive supply voltage via a current source or pull-up resistor (see Figure 30-2 on page 363). When the bus is free, both lines are high. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function.

TWD and TWCK pins may be multiplexed with PIO lines. To enable the TWI, the programmer must perform the following steps:

- Program the PIO controller to:
  - Dedicate TWD and TWCK as peripheral lines.
  - Define TWD and TWCK as open-drain.

30.4.2 Power Management

- Enable the peripheral clock.

The TWI interface may be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the TWI clock.

30.4.3 Interrupt

The TWI interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). In order to handle interrupts, the AIC must be programmed before configuring the TWI.

30.5 Functional Description

30.5.1 Transfer Format

The data put on the TWD line must be 8 bits long. Data is transferred MSB first; each byte must be followed by an acknowledgement. The number of bytes per transfer is unlimited (see Figure 30-4 on page 365).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 30-3 on page 365).

- A high-to-low transition on the TWD line while TWCK is high defines the START condition.
- A low-to-high transition on the TWD line while TWCK is high defines a STOP condition.

Figure 30-3. START and STOP Conditions

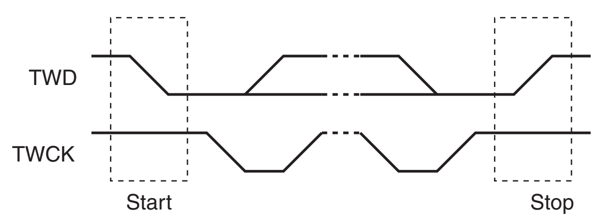
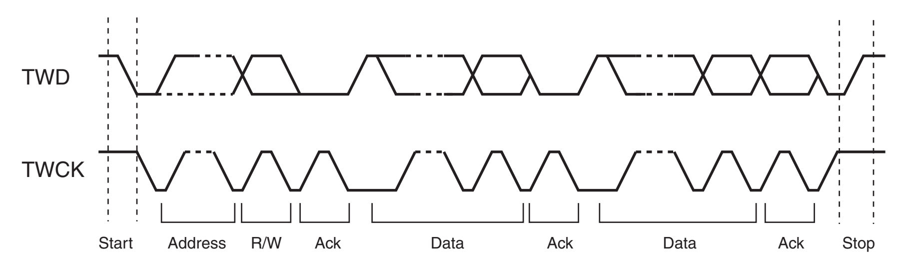


Figure 30-4. Transfer Format



30.5.2 Modes of Operation

The TWI has two modes of operation:

- Master transmitter mode
- Master receiver mode

The TWI Control Register (TWI_CR) allows configuration of the interface in Master Mode. In this mode, it generates the clock according to the value programmed in the Clock Waveform Generator Register (TWI_CWGR). This register defines the TWCK signal completely, enabling the interface to be adapted to a wide range of clocks.

30.5.3 Transmitting Data

After the master initiates a Start condition, it sends a 7-bit slave address, configured in the Master Mode register (DADR in TWI_MMR), to notify the slave device. The bit following the slave address indicates the transfer direction (write or read). If this bit is 0, it indicates a write operation (transmit operation). If the bit is 1, it indicates a request for data read (receive operation).

The TWI transfers require the slave to acknowledge each received byte. During the acknowledge clock pulse, the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. The master polls the data line during this clock pulse and

sets the **NAK** bit in the status register if the slave does not acknowledge the byte. As with the other status bits, an interrupt can be generated if enabled in the interrupt enable register (TWI_IER). After writing in the transmit-holding register (TWI_THR), setting the START bit in the control register starts the transmission. The data is shifted in the internal shifter and when an acknowledge is detected, the TXRDY bit is set until a new write in the TWI_THR (see Figure 30-6 below). The master generates a stop condition to end the transfer.

The read sequence begins by setting the START bit. When the RXRDY bit is set in the status register, a character has been received in the receive-holding register (TWI_RHR). The RXRDY bit is reset when reading the TWI_RHR.

The TWI interface performs various transfer formats (7-bit slave address, 10-bit slave address). The three internal address bytes are configurable through the Master Mode register (TWI_MMR). If the slave device supports only a 7-bit address, **IADRSZ** must be set to 0. For a slave address higher than 7 bits, the user must configure the address size **(IADRSZ)** and set the other slave address bits in the internal address register (TWI_IADR).

Figure 30-5. Master Write with One, Two or Three Bytes Internal Address and One Data Byte

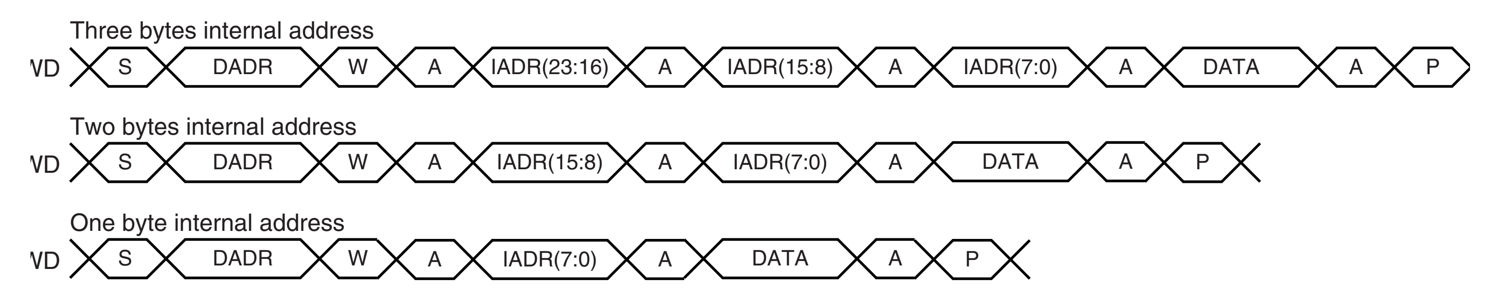


Figure 30-6. Master Write with One Byte Internal Address and Multiple Data Bytes

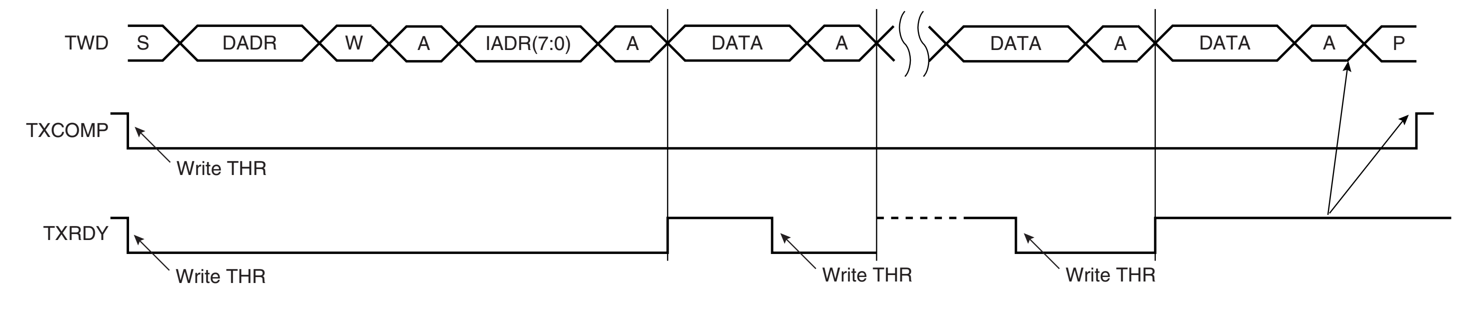


Figure 30-7. Master Read with One, Two or Three Bytes Internal Address and One Data Byte

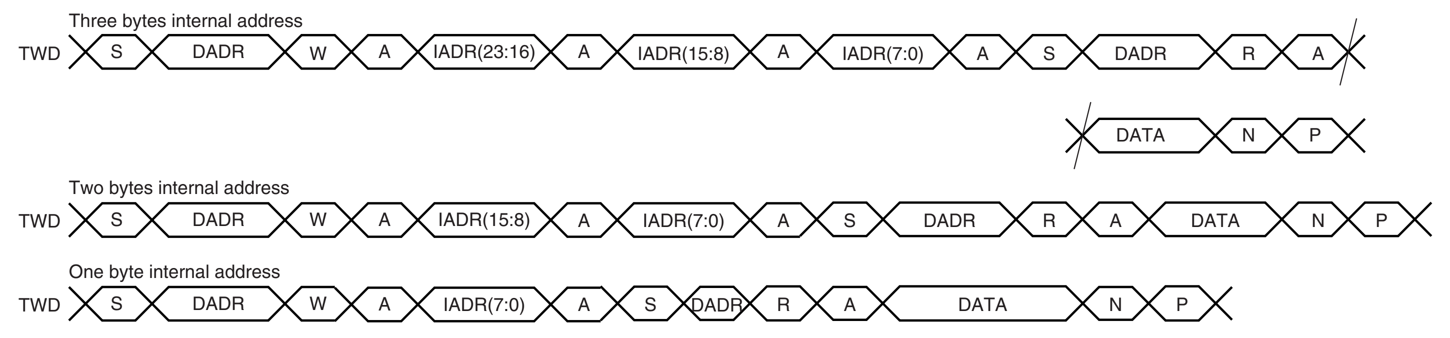
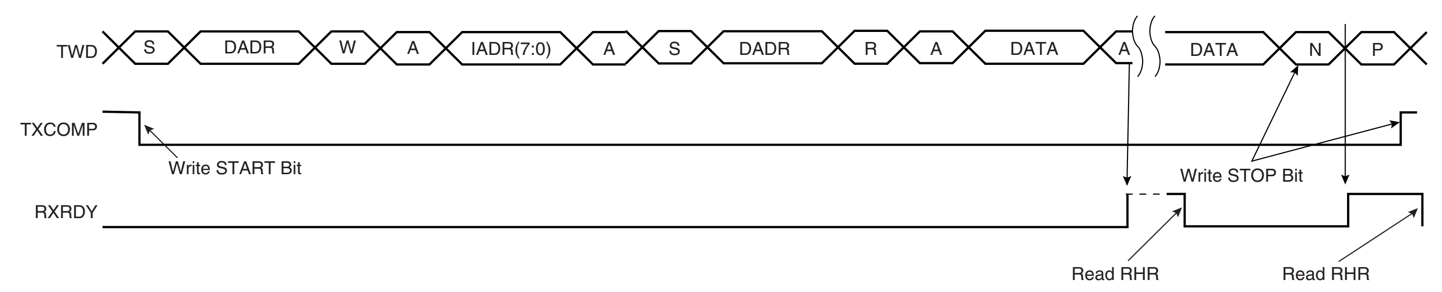


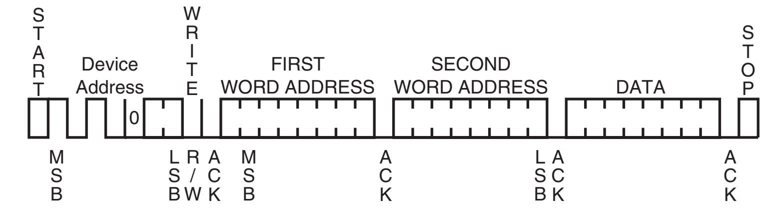
Figure 30-8. Master Read with One Byte Internal Address and Multiple Data Bytes



- S = Start
- P = Stop
- W = Write
- R = Read
- A = Acknowledge
- N = Not Acknowledge
- DADR = Device Address
- IADR = Internal Address

Figure 30-9 below shows a byte write to an Atmel AT24LC512 EEPROM. This demonstrates the use of internal addresses to access the device.

Figure 30-9. Internal Address Usage



30.5.4 Read/Write Flowcharts

The following flowcharts shown in Figure 30-10 on page 368 and in Figure 30-11 on page 369 give examples for read and write operations in Master Mode. A polling or interrupt method can be used to check the status bits. The interrupt method requires that the interrupt enable register (TWI_IER) be configured first.

Figure 30-10. TWI Write in Master Mode

START

Set TWI clock:
TWI_CWGR = clock

Set the control register:
- Master enable
TWI_CR = MSEN

Set the Master Mode register:
- Device slave address
- Internal address size
- Transfer direction bit
Write ==> bit MREAD = 0

Internal address size = 0?

Set theinternal address
TWI_IADR = address

Yes

Load transmit register
TWI_THR = Data to send
Start the transfer
TWI_CR = START

Read status register

TWI_THR = data to send

TXRDY = 0?

Yes

Data to send?

Yes

Stop the transfer
TWI_CR = STOP

Read status register

TXCOMP = 0?

Yes

END

Figure 30-11. TWI Read in Master Mode

30.6 Two-wire Interface (TWI) User Interface

Table 30-2. Two-wire Interface (TWI) Register Mapping

Offset	Register	Name	Access	Reset Value
0x0000	Control Register	TWI_CR	Write-only	N/A
0x0004	Master Mode Register	TWI_MMR	Read/Write	0x0000
0x0008	Reserved	–	–	–
0x000C	Internal Address Register	TWI_IADR	Read/Write	0x0000
0x0010	Clock Waveform Generator Register	TWI_CWGR	Read/Write	0x0000
0x0020	Status Register	TWI_SR	Read-only	0x0008
0x0024	Interrupt Enable Register	TWI_IER	Write-only	N/A
0x0028	Interrupt Disable Register	TWI_IDR	Write-only	N/A
0x002C	Interrupt Mask Register	TWI_IMR	Read-only	0x0000
0x0030	Receive Holding Register	TWI_RHR	Read-only	0x0000
0x0034	Transmit Holding Register	TWI_THR	Read/Write	0x0000
0x0038-0x00FC	Reserved	–	–	–

30.6.1 TWI Control Register

Register Name: TWI_CR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SWRST	–	–	–	MSDIS	MSEN	STOP	START

- **START: Send a START Condition**

0 = No effect.

1 = A frame beginning with a START bit is transmitted according to the features defined in the mode register.

This action is necessary when the TWI peripheral wants to read data from a slave. When configured in Master Mode with a write operation, a frame is sent with the mode register as soon as the user writes a character in the holding register.

- **STOP: Send a STOP Condition**

0 = No effect.

1 = STOP Condition is sent just after completing the current byte transmission in master read or write mode.

In single data byte master read or write, the START and STOP must both be set.

In multiple data bytes master read or write, the STOP must be set before ACK/NACK bit transmission.

In master read mode, if a NACK bit is received, the STOP is automatically performed.

In multiple data write operation, when both THR and shift register are empty, a STOP condition is automatically sent.

- **MSEN: TWI Master Transfer Enabled**

0 = No effect.

1 = If MSDIS = 0, the master data transfer is enabled.

- **MSDIS: TWI Master Transfer Disabled**

0 = No effect.

1 = The master data transfer is disabled, all pending data is transmitted. The shifter and holding characters (if they contain data) are transmitted in case of write operation. In read operation, the character being transferred must be completely received before disabling.

- **SWRST: Software Reset**

0 = No effect.

1 = Equivalent to a system reset.



30.6.2 TWI Master Mode Register

Register Name: TWI_MMR

Address Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	DADR						

15	14	13	12	11	10	9	8
–	–	–	MREAD	–	–	IADRSZ	

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **IADRSZ: Internal Device Address Size**

IADRSZ[9:8]		
0	0	No internal device address (Byte command protocol)
0	1	One-byte internal device address
1	0	Two-byte internal device address
1	1	Three-byte internal device address

- **MREAD: Master Read Direction**

0 = Master write direction.

1 = Master read direction.

- **DADR: Device Address**

The device address is used in Master Mode to access slave devices in read or write mode.

30.6.3 TWI Internal Address Register

Register Name: TWI_IADR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
IADR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
IADR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IADR							

- **IADR: Internal Address**

0, 1, 2 or 3 bytes depending on IADRSZ.

– Low significant byte address in 10-bit mode addresses.

30.6.4 TWI Clock Waveform Generator Register

Register Name: TWI_CWGR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	CKDIV		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CHDIV							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CLDIV							

- **CLDIV: Clock Low Divider**

The SCL low period is defined as follows:

$$T_{low} = ((\text{CLDIV} \times 2^{\text{CKDIV}}) + 3) \times T_{MCK}$$

- **CHDIV: Clock High Divider**

The SCL high period is defined as follows:

$$T_{high} = ((\text{CHDIV} \times 2^{\text{CKDIV}}) + 3) \times T_{MCK}$$

- **CKDIV: Clock Divider**

The CKDIV is used to increase both SCL high and low periods.

30.6.5 TWI Status Register

Register Name: TWI_SR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	NACK
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UNRE	OVRE	–	–	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed**

0 = In master, during the length of the current frame. In slave, from START received to STOP received.

1 = When both holding and shift registers are empty and STOP condition has been sent (in Master) or when MSEN is set (enable TWI).

- **RXRDY: Receive Holding Register Ready**

0 = No character has been received since the last TWI_RHR read operation.

1 = A byte has been received in the TWI_RHR since the last read.

- **TXRDY: Transmit Holding Register Ready**

0 = The transmit holding register has not been transferred into shift register. Set to 0 when writing into TWI_THR register.

1 = As soon as data byte is transferred from TWI_THR to internal shifter or if a NACK error is detected, TXRDY is set at the same time as TXCOMP and NACK. TXRDY is also set when MSEN is set (enable TWI).

- **OVRE: Overrun Error**

0 = TWI_RHR has not been loaded while RXRDY was set

1 = TWI_RHR has been loaded while RXRDY was set. Reset by read in TWI_SR when TXCOMP is set.

- **UNRE: Underrun Error**

0 = No underrun error

1 = No valid data in TWI_THR (TXRDY set) while trying to load the data shifter. This action automatically generated a STOP bit in Master Mode. Reset by read in TWI_SR when TXCOMP is set.

- **NACK: Not Acknowledged**

0 = Each data byte has been correctly received by the far-end side TWI slave component.

1 = A data byte has not been acknowledged by the slave component. Set at the same time as TXCOMP. Reset after read.

30.6.6 TWI Interrupt Enable Register

Register Name: TWI_IER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	NACK
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UNRE	OVRE	–	–	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed**
- **RXRDY: Receive Holding Register Ready**
- **TXRDY: Transmit Holding Register Ready**
- **OVRE: Overrun Error**
- **UNRE: Underrun Error**
- **NACK: Not Acknowledge**

0 = No effect.

1 = Enables the corresponding interrupt.

30.6.7 TWI Interrupt Disable Register

Register Name: TWI_IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	NACK
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UNRE	OVRE	–	–	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed**

- **RXRDY: Receive Holding Register Ready**

- **TXRDY: Transmit Holding Register Ready**

- **OVRE: Overrun Error**

- **UNRE: Underrun Error**

- **NACK: Not Acknowledge**

0 = No effect.

1 = Disables the corresponding interrupt.

30.6.8 TWI Interrupt Mask Register

Register Name: TWI_IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	NACK
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UNRE	OVRE	–	–	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed**

- **RXRDY: Receive Holding Register Ready**

- **TXRDY: Transmit Holding Register Ready**

- **OVRE: Overrun Error**

- **UNRE: Underrun Error**

- **NACK: Not Acknowledge**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

30.6.9 TWI Receive Holding Register

Register Name: TWI_RHR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXDATA							

- **RXDATA: Master or Slave Receive Holding Data**

30.6.10 TWI Transmit Holding Register

Register Name: TWI_THR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXDATA							

- **TXDATA: Master or Slave Transmit Holding Data**

31. Universal Synchronous/Asynchronous Receiver/Transmitter

31.1 Description

The Universal Synchronous Asynchronous Receiver Transceiver (USART) provides one full duplex universal synchronous asynchronous serial link. Data frame format is widely programmable (data length, parity, number of stop bits) to support a maximum of standards. The receiver implements parity error, framing error and overrun error detection. The receiver time-out enables handling variable-length frames and the transmitter timeguard facilitates communications with slow remote devices. Multidrop communications are also supported through address bit handling in reception and transmission.

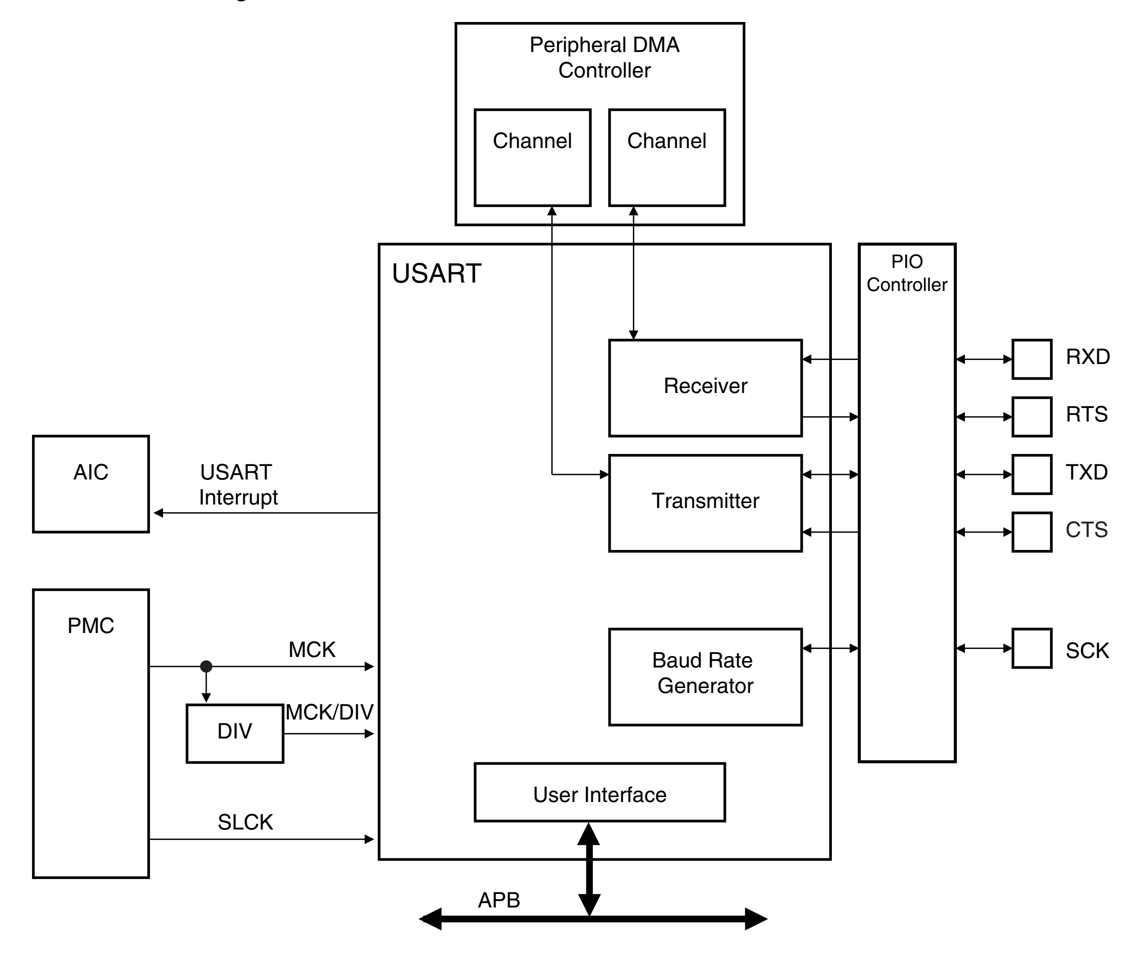
The USART features three test modes: remote loopback, local loopback and automatic echo.

The USART supports specific operating modes providing interfaces on RS485 buses, with ISO7816 T = 0 or T = 1 smart card slots and infrared transceivers. The hardware handshaking feature enables an out-of-band flow control by automatic management of the pins RTS and CTS.

The USART supports the connection to the Peripheral DMA Controller, which enables data transfers to the transmitter and from the receiver. The PDC provides chained buffer management without any intervention of the processor.

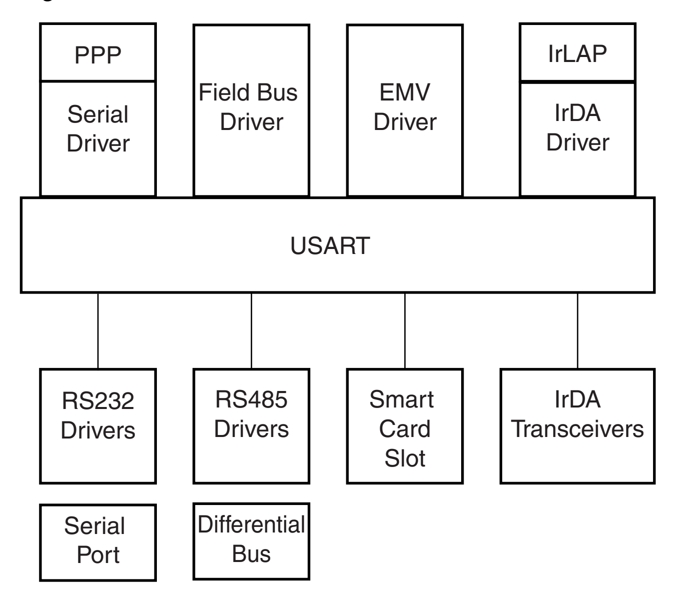
31.2 Block Diagram

Figure 31-1. USART Block Diagram



31.3 Application Block Diagram

Figure 31-2. Application Block Diagram



31.4 I/O Lines Description

Table 31-1. I/O Line Description

Name	Description	Type	Active Level
SCK	Serial Clock	I/O	
TXD	Transmit Serial Data	I/O	
RXD	Receive Serial Data	Input	
CTS	Clear to Send	Input	Low
RTS	Request to Send	Output	Low

31.5 Product Dependencies

31.5.1 I/O Lines

The pins used for interfacing the USART may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the desired USART pins to their peripheral function. If I/O lines of the USART are not used by the application, they can be used for other purposes by the PIO Controller.

31.5.2 Power Management

The USART is not continuously clocked. The programmer must first enable the USART Clock in the Power Management Controller (PMC) before using the USART. However, if the application does not require USART operations, the USART clock can be stopped when not needed and be restarted later. In this case, the USART will resume its operations where it left off.

Configuring the USART does not require the USART clock to be enabled.

31.5.3 Interrupt

The USART interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the USART interrupt requires the AIC to be programmed first. Note that it is not recommended to use the USART interrupt line in edge sensitive mode.

31.6 Functional Description

The USART is capable of managing several types of serial synchronous or asynchronous communications.

It supports the following communication modes:

- 5- to 9-bit full-duplex asynchronous serial communication
  - MSB- or LSB-first
  - 1, 1.5 or 2 stop bits
  - Parity even, odd, marked, space or none
  - By 8 or by 16 over-sampling receiver frequency
  - Optional hardware handshaking
  - Optional break management
  - Optional multidrop serial communication
- High-speed 5- to 9-bit full-duplex synchronous serial communication
  - MSB- or LSB-first
  - 1 or 2 stop bits
  - Parity even, odd, marked, space or none
  - By 8 or by 16 over-sampling frequency
  - Optional hardware handshaking
  - Optional break management
  - Optional multidrop serial communication
- RS485 with driver control signal
- ISO7816, T0 or T1 protocols for interfacing with smart cards
  - NACK handling, error counter with repetition and iteration limit
- InfraRed IrDA Modulation and Demodulation
- Test modes
  - Remote loopback, local loopback, automatic echo

31.6.1 Baud Rate Generator

The Baud Rate Generator provides the bit period clock named the Baud Rate Clock to both the receiver and the transmitter.

The Baud Rate Generator clock source can be selected by setting the USCLKS field in the Mode Register (US_MR) between:

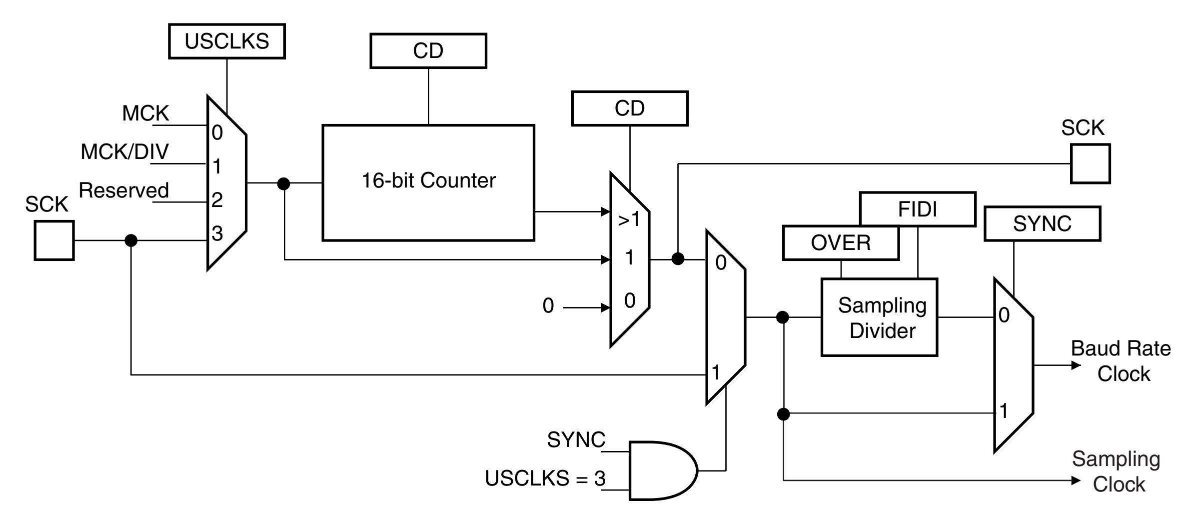
- the Master Clock MCK
- a division of the Master Clock, the divider being product dependent, but generally set to 8
- the external clock, available on the SCK pin

The Baud Rate Generator is based upon a 16-bit divider, which is programmed with the CD field of the Baud Rate Generator Register (US_BRGR). If CD is programmed at 0, the Baud Rate Generator does not generate any clock. If CD is programmed at 1, the divider is bypassed and becomes inactive.

If the external SCK clock is selected, the duration of the low and high levels of the signal provided on the SCK pin must be longer than a Master Clock (MCK) period. The frequency of the signal provided on SCK must be at least 4.5 times lower than MCK.



Figure 31-3. Baud Rate Generator



31.6.1.1 *Baud Rate in Asynchronous Mode*

If the USART is programmed to operate in asynchronous mode, the selected clock is first divided by CD, which is field programmed in the Baud Rate Generator Register (US_BRGR). The resulting clock is provided to the receiver as a sampling clock and then divided by 16 or 8, depending on the programming of the OVER bit in US_MR.

If OVER is set to 1, the receiver sampling is 8 times higher than the baud rate clock. If OVER is cleared, the sampling is performed at 16 times the baud rate clock.

The following formula performs the calculation of the Baud Rate.

$$Baudrate = \frac{SelectedClock}{(8(2 - Over)CD)}$$

This gives a maximum baud rate of MCK divided by 8, assuming that MCK is the highest possible clock and that OVER is programmed at 1.

- Baud Rate Calculation Example

Table 31-2 shows calculations of CD to obtain a baud rate at 38400 bauds for different source clock frequencies. This table also shows the actual resulting baud rate and the error.

Table 31-2. Baud Rate Example (OVER = 0)

Source Clock	Expected Baud Rate	Calculation Result	CD	Actual Baud Rate	Error
MHz	Bit/s			Bit/s	
3 686 400	38 400	6.00	6	38 400.00	0.00%
4 915 200	38 400	8.00	8	38 400.00	0.00%
5 000 000	38 400	8.14	8	39 062.50	1.70%
7 372 800	38 400	12.00	12	38 400.00	0.00%
8 000 000	38 400	13.02	13	38 461.54	0.16%
12 000 000	38 400	19.53	20	37 500.00	2.40%
12 288 000	38 400	20.00	20	38 400.00	0.00%



Table 31-2. Baud Rate Example (OVER = 0) (Continued)

Source Clock	Expected Baud Rate	Calculation Result	CD	Actual Baud Rate	Error
14 318 180	38 400	23.30	23	38 908.10	1.31%
14 745 600	38 400	24.00	24	38 400.00	0.00%
18 432 000	38 400	30.00	30	38 400.00	0.00%
24 000 000	38 400	39.06	39	38 461.54	0.16%
24 576 000	38 400	40.00	40	38 400.00	0.00%
25 000 000	38 400	40.69	40	38 109.76	0.76%
32 000 000	38 400	52.08	52	38 461.54	0.16%
32 768 000	38 400	53.33	53	38 641.51	0.63%
33 000 000	38 400	53.71	54	38 194.44	0.54%
40 000 000	38 400	65.10	65	38 461.54	0.16%
50 000 000	38 400	81.38	81	38 580.25	0.47%
60 000 000	38 400	97.66	98	38 265.31	0.35%
70 000 000	38 400	113.93	114	38 377.19	0.06%

The baud rate is calculated with the following formula:

$$BaudRate = MCK / CD \times 16$$

The baud rate error is calculated with the following formula. It is not recommended to work with an error higher than 5%.

$$Error = 1 - \left(\frac{ExpectedBaudRate}{ActualBaudRate}\right)$$

31.6.1.2 Baud Rate in Synchronous Mode

If the USART is programmed to operate in synchronous mode, the selected clock is simply divided by the field CD in US_BRGR.

$$BaudRate = \frac{SelectedClock}{CD}$$

In synchronous mode, if the external clock is selected (USCLKS = 3), the clock is provided directly by the signal on the USART SCK pin. No division is active. The value written in US_BRGR has no effect. The external clock frequency must be at least 4.5 times lower than the system clock.

When either the external clock SCK or the internal clock divided (MCK/DIV) is selected, the value programmed in CD must be even if the user has to ensure a 50:50 mark/space ratio on the SCK pin. If the internal clock MCK is selected, the Baud Rate Generator ensures a 50:50 duty cycle on the SCK pin, even if the value programmed in CD is odd.

31.6.1.3 Baud Rate in ISO 7816 Mode

The ISO7816 specification defines the bit rate with the following formula:

$$B = \frac{Di}{Fi} \times f$$

where:

- B is the bit rate
- Di is the bit-rate adjustment factor
- Fi is the clock frequency division factor
- f is the ISO7816 clock frequency (Hz)

Di is a binary value encoded on a 4-bit field, named DI, as represented in Table 31-3.

Table 31-3. Binary and Decimal Values for D

DI field	0001	0010	0011	0100	0101	0110	1000	1001
Di (decimal)	1	2	4	8	16	32	12	20

Fi is a binary value encoded on a 4-bit field, named FI, as represented in Table 31-4.

Table 31-4. Binary and Decimal Values for F

FI field	0000	0001	0010	0011	0100	0101	0110	1001	1010	1011	1100	1101
Fi (decimal	372	372	558	744	1116	1488	1860	512	768	1024	1536	2048

Table 31-5 shows the resulting Fi/Di Ratio, which is the ratio between the ISO7816 clock and the baud rate clock.

Table 31-5. Possible Values for the Fi/Di Ratio

Fi/Di	372	558	774	1116	1488	1806	512	768	1024	1536	2048
1	372	558	744	1116	1488	1860	512	768	1024	1536	2048
2	186	279	372	558	744	930	256	384	512	768	1024
4	93	139.5	186	279	372	465	128	192	256	384	512
8	46.5	69.75	93	139.5	186	232.5	64	96	128	192	256
16	23.25	34.87	46.5	69.75	93	116.2	32	48	64	96	128
32	11.62	17.43	23.25	34.87	46.5	58.13	16	24	32	48	64
12	31	46.5	62	93	124	155	42.66	64	85.33	128	170.6
20	18.6	27.9	37.2	55.8	74.4	93	25.6	38.4	51.2	76.8	102.4

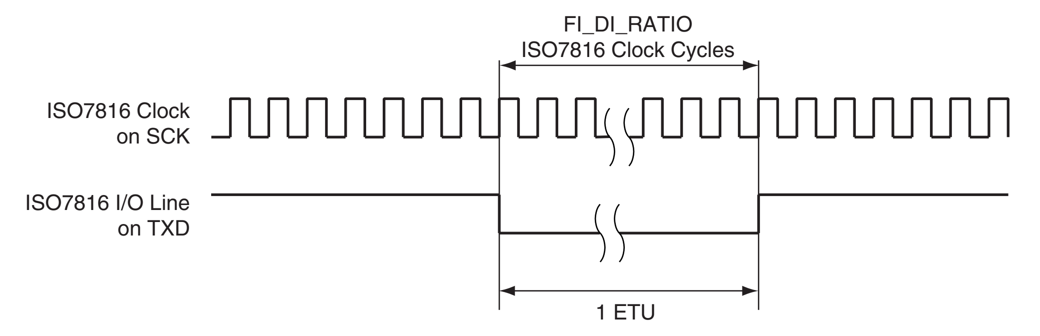
If the USART is configured in ISO7816 Mode, the clock selected by the USCLKS field in the Mode Register (US_MR) is first divided by the value programmed in the field CD in the Baud Rate Generator Register (US_BRGR). The resulting clock can be provided to the SCK pin to feed the smart card clock inputs. This means that the CLKO bit can be set in US_MR.

This clock is then divided by the value programmed in the FI_DI_RATIO field in the FI_DI_Ratio register (US_FIDI). This is performed by the Sampling Divider, which performs a division by up to 2047 in ISO7816 Mode. The non-integer values of the Fi/Di Ratio are not supported and the user must program the FI_DI_RATIO field to a value as close as possible to the expected value.

The FI_DI_RATIO field resets to the value 0x174 (372 in decimal) and is the most common divider between the ISO7816 clock and the bit rate (Fi = 372, Di = 1).

Figure 31-4 shows the relation between the Elementary Time Unit, corresponding to a bit time, and the ISO 7816 clock.

Figure 31-4. Elementary Time Unit (ETU)



31.6.2 Receiver and Transmitter Control

After reset, the receiver is disabled. The user must enable the receiver by setting the RXEN bit in the Control Register (US_CR). However, the receiver registers can be programmed before the receiver clock is enabled.

After reset, the transmitter is disabled. The user must enable it by setting the TXEN bit in the Control Register (US_CR). However, the transmitter registers can be programmed before being enabled.

The Receiver and the Transmitter can be enabled together or independently.

At any time, the software can perform a reset on the receiver or the transmitter of the USART by setting the corresponding bit, RSTRX and RSTTX respectively, in the Control Register (US_CR). The reset commands have the same effect as a hardware reset on the corresponding logic. Regardless of what the receiver or the transmitter is performing, the communication is immediately stopped.

The user can also independently disable the receiver or the transmitter by setting RXDIS and TXDIS respectively in US_CR. If the receiver is disabled during a character reception, the USART waits until the end of reception of the current character, then the reception is stopped. If the transmitter is disabled while it is operating, the USART waits the end of transmission of both the current character and character being stored in the Transmit Holding Register (US_THR). If a timeguard is programmed, it is handled normally.

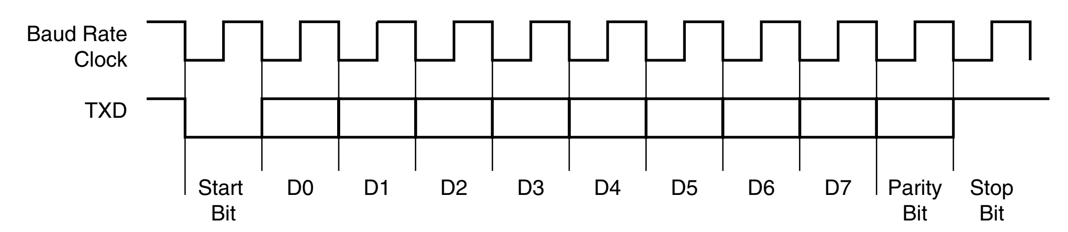
31.6.3 Synchronous and Asynchronous Modes

31.6.3.1 Transmitter Operations

The transmitter performs the same in both synchronous and asynchronous operating modes (SYNC = 0 or SYNC = 1). One start bit, up to 9 data bits, one optional parity bit and up to two stop bits are successively shifted out on the TXD pin at each falling edge of the programmed serial clock.

The number of data bits is selected by the CHRL field and the MODE9 bit in the Mode Register (US_MR). Nine bits are selected by setting the MODE 9 bit regardless of the CHRL field. The parity bit is set according to the PAR field in US_MR. The even, odd, space, marked or none parity bit can be configured. The MSBF field in US_MR configures which data bit is sent first. If written at 1, the most significant bit is sent first. At 0, the less significant bit is sent first. The number of stop bits is selected by the NBSTOP field in US_MR. The 1.5 stop bit is supported in asynchronous mode only.

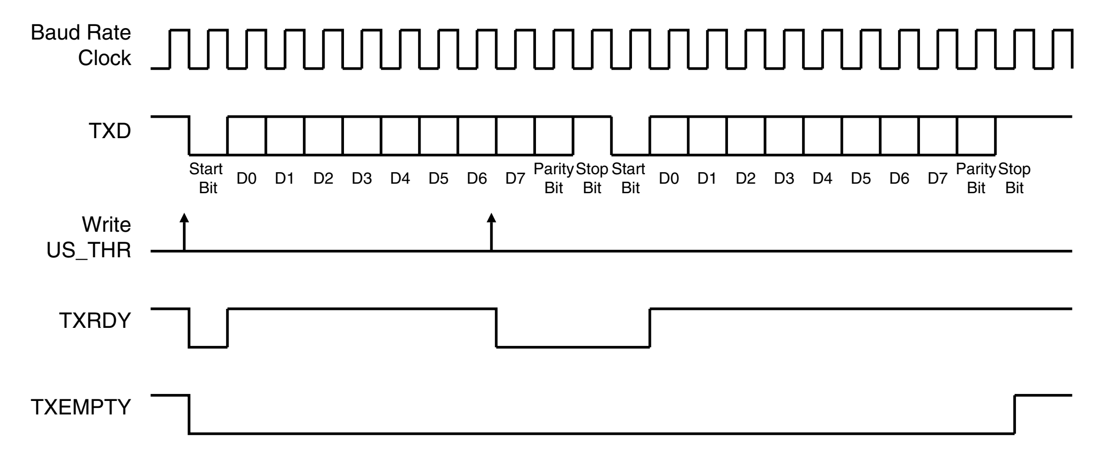
Figure 31-5. Character Transmit



The characters are sent by writing in the Transmit Holding Register (US_THR). The transmitter reports two status bits in the Channel Status Register (US_CSR): TXRDY (Transmitter Ready), which indicates that US_THR is empty and TXEMPTY, which indicates that all the characters written in US_THR have been processed. When the current character processing is completed, the last character written in US_THR is transferred into the Shift Register of the transmitter and US_THR becomes empty, thus TXRDY raises.

Both TXRDY and TXEMPTY bits are low since the transmitter is disabled. Writing a character in US_THR while TXRDY is active has no effect and the written character is lost.

Figure 31-6. Transmitter Status



31.6.3.2 *Asynchronous Receiver*

If the USART is programmed in asynchronous operating mode (SYNC = 0), the receiver oversamples the RXD input line. The oversampling is either 16 or 8 times the Baud Rate clock, depending on the OVER bit in the Mode Register (US_MR).

The receiver samples the RXD line. If the line is sampled during one half of a bit time at 0, a start bit is detected and data, parity and stop bits are successively sampled on the bit rate clock.

If the oversampling is 16, (OVER at 0), a start is detected at the eighth sample at 0. Then, data bits, parity bit and stop bit are sampled on each 16 sampling clock cycle. If the oversampling is 8 (OVER at 1), a start bit is detected at the fourth sample at 0. Then, data bits, parity bit and stop bit are sampled on each 8 sampling clock cycle.

The number of data bits, first bit sent and parity mode are selected by the same fields and bits as the transmitter, i.e. respectively CHRL, MODE9, MSBF and PAR. The number of stop bits has no effect on the receiver as it considers only one stop bit, regardless of the field NBSTOP,

so that resynchronization between the receiver and the transmitter can occur. Moreover, as soon as the stop bit is sampled, the receiver starts looking for a new start bit so that resynchronization can also be accomplished when the transmitter is operating with one stop bit.

Figure 31-7 and Figure 31-8 illustrate start detection and character reception when USART operates in asynchronous mode.

Figure 31-7. Asynchronous Start Detection

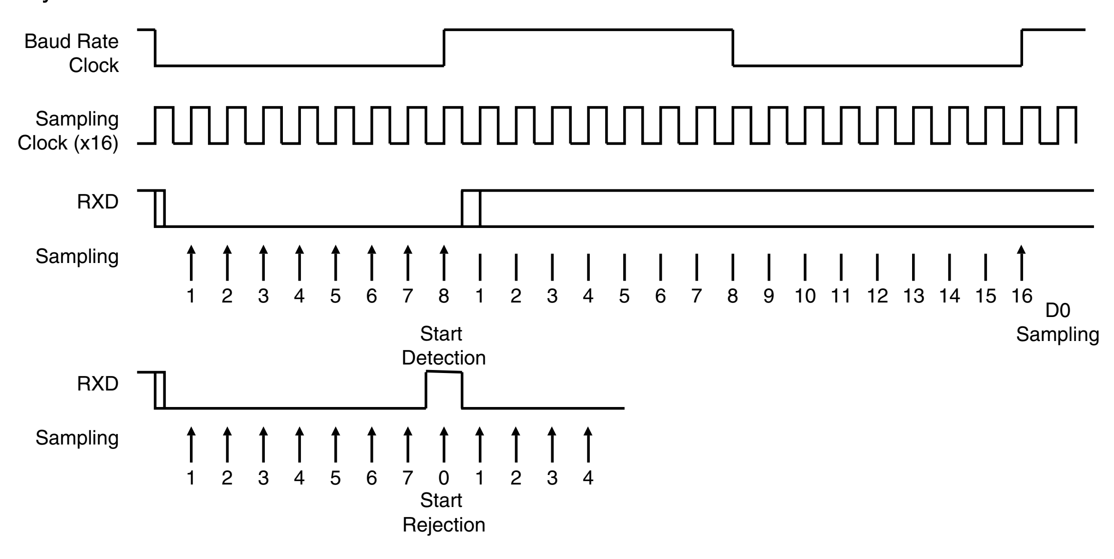
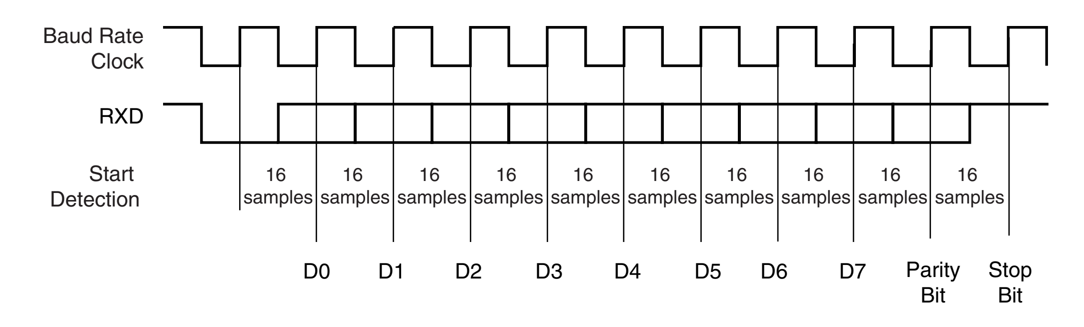


Figure 31-8. Asynchronous Character Reception



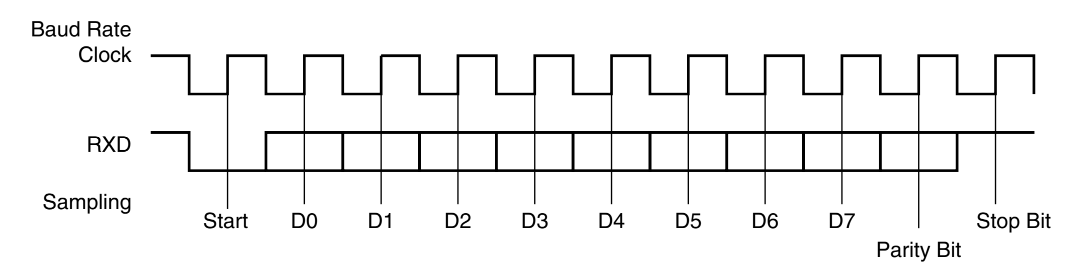
31.6.3.3 *Synchronous Receiver*

In synchronous mode (SYNC = 1), the receiver samples the RXD signal on each rising edge of the Baud Rate Clock. If a low level is detected, it is considered as a start. All data bits, the parity bit and the stop bits are sampled and the receiver waits for the next start bit. Synchronous mode operations provide a high speed transfer capability.

Configuration fields and bits are the same as in asynchronous mode.

Figure 31-9 illustrates a character reception in synchronous mode.

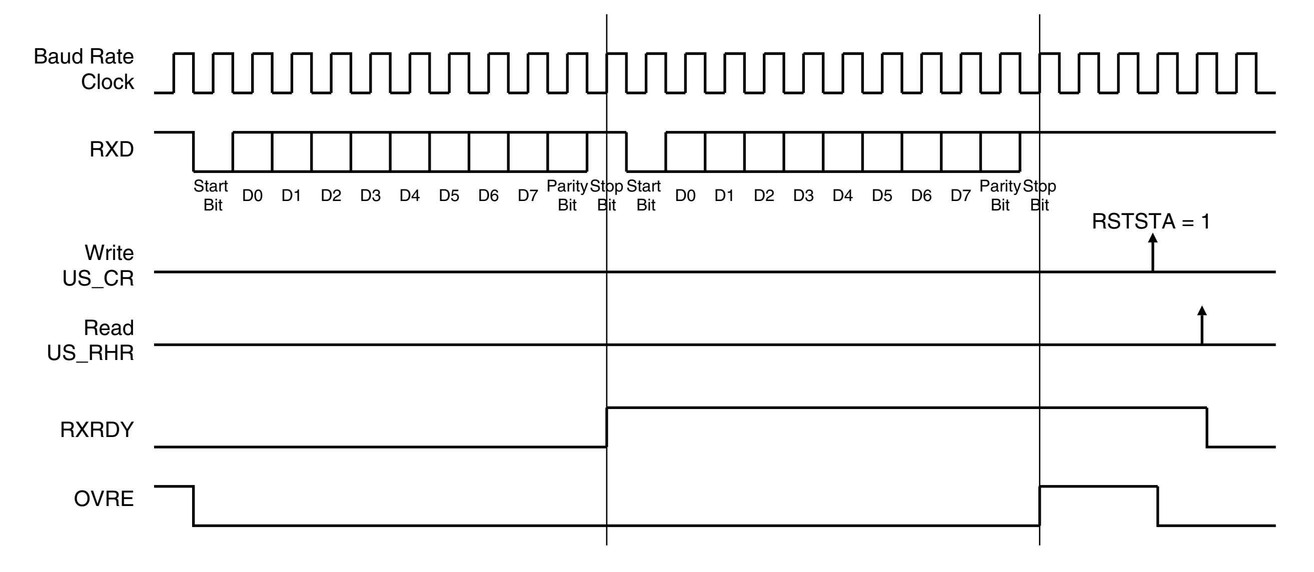
Figure 31-9. Synchronous Mode Character Reception



31.6.3.4 Receiver Operations

When a character reception is completed, it is transferred to the Receive Holding Register (US_RHR) and the RXRDY bit in the Status Register (US_CSR) rises. If a character is completed while the RXRDY is set, the OVRE (Overrun Error) bit is set. The last character is transferred into US_RHR and overwrites the previous one. The OVRE bit is cleared by writing the Control Register (US_CR) with the RSTSTA (Reset Status) bit at 1.

Figure 31-10. Receiver Status



31.6.3.5 Parity

The USART supports five parity modes selected by programming the PAR field in the Mode Register (US_MR). The PAR field also enables the Multidrop mode, see "Multidrop Mode" on page 392. Even and odd parity bit generation and error detection are supported.

If even parity is selected, the parity generator of the transmitter drives the parity bit at 0 if a number of 1s in the character data bit is even, and at 1 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If odd parity is selected, the parity generator of the transmitter drives the parity bit at 1 if a number of 1s in the character data bit is even, and at 0 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If the mark parity is used, the parity generator of the transmitter drives the parity bit at 1 for all characters. The receiver parity checker reports an error if the parity bit is sampled at 0. If the space parity is used, the parity generator of the transmitter drives the parity bit at 0 for all characters. The receiver parity checker reports an error if the parity bit is sampled at 1. If parity is disabled, the transmitter does not generate any parity bit and the receiver does not report any parity error.

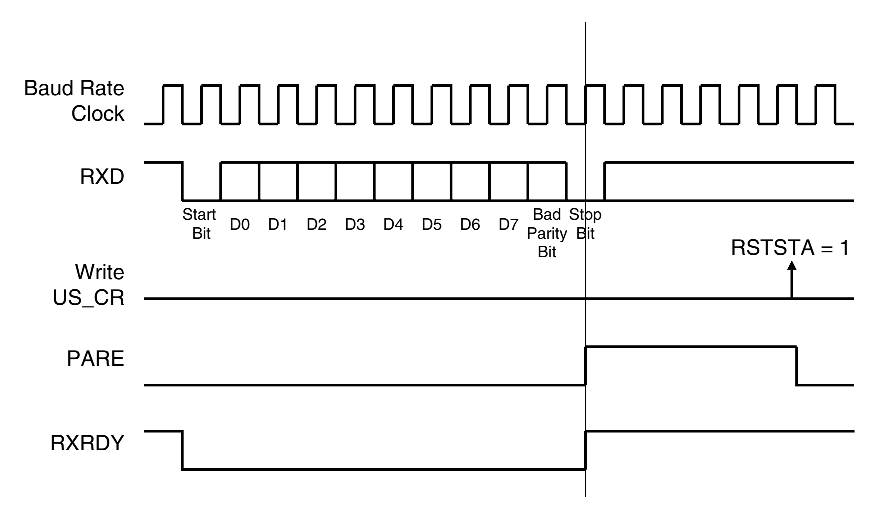
Table 31-6 shows an example of the parity bit for the character 0x41 (character ASCII "A") depending on the configuration of the USART. Because there are two bits at 1, 1 bit is added when a parity is odd, or 0 is added when a parity is even.

Table 31-6. Parity Bit Examples

Character	Hexa	Binary	Parity Bit	Parity Mode
A	0x41	0100 0001	1	Odd
A	0x41	0100 0001	0	Even
A	0x41	0100 0001	1	Mark
A	0x41	0100 0001	0	Space
A	0x41	0100 0001	None	None

When the receiver detects a parity error, it sets the PARE (Parity Error) bit in the Channel Status Register (US_CSR). The PARE bit can be cleared by writing the Control Register (US_CR) with the RSTSTA bit at 1. Figure 31-11 illustrates the parity bit status setting and clearing.

Figure 31-11. Parity Error



31.6.3.6 *Multidrop Mode*

If the PAR field in the Mode Register (US_MR) is programmed to the value 0x6 or 0x07, the USART runs in Multidrop Mode. This mode differentiates the data characters and the address characters. Data is transmitted with the parity bit at 0 and addresses are transmitted with the parity bit at 1.

If the USART is configured in multidrop mode, the receiver sets the PARE parity error bit when the parity bit is high and the transmitter is able to send a character with the parity bit high when the Control Register is written with the SENDA bit at 1.

To handle parity error, the PARE bit is cleared when the Control Register is written with the bit RSTSTA at 1.

The transmitter sends an address byte (parity bit set) when SENDA is written to US_CR. In this case, the next byte written to US_THR is transmitted as an address. Any character written in US_THR without having written the command SENDA is transmitted normally with the parity at 0.

31.6.3.7 *Transmitter Timeguard*

The timeguard feature enables the USART interface with slow remote devices.

The timeguard function enables the transmitter to insert an idle state on the TXD line between two characters. This idle state actually acts as a long stop bit.

The duration of the idle state is programmed in the TG field of the Transmitter Timeguard Register (US_TTGR). When this field is programmed at zero no timeguard is generated. Otherwise, the transmitter holds a high level on TXD after each transmitted byte during the number of bit periods programmed in TG in addition to the number of stop bits.

As illustrated in Figure 31-12, the behavior of TXRDY and TXEMPTY status bits is modified by the programming of a timeguard. TXRDY rises only when the start bit of the next character is sent, and thus remains at 0 during the timeguard transmission if a character has been written in US_THR. TXEMPTY remains low until the timeguard transmission is completed as the timeguard is part of the current character being transmitted.

Figure 31-12. Timeguard Operations

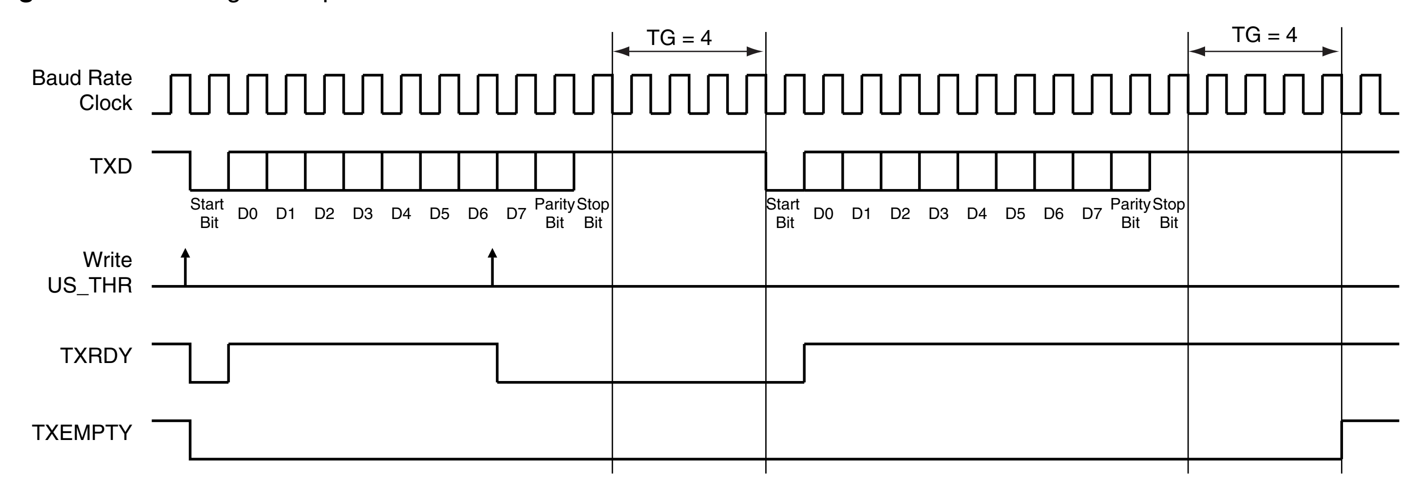


Table 31-7 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the Baud Rate.

Table 31-7. Maximum Timeguard Length Depending on Baud Rate

Baud Rate	Bit time	Timeguard
Bit/sec	µs	ms
1 200	833	212.50
9 600	104	26.56
14400	69.4	17.71
19200	52.1	13.28
28800	34.7	8.85
33400	29.9	7.63
56000	17.9	4.55
57600	17.4	4.43
115200	8.7	2.21

31.6.3.8 Receiver Time-out

The Receiver Time-out provides support in handling variable-length frames. This feature detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in the Channel Status Register (US_CSR) rises and can generate an interrupt, thus indicating to the driver an end of frame.

The time-out delay period (during which the receiver waits for a new character) is programmed in the TO field of the Receiver Time-out Register (US_RTOR). If the TO field is programmed at 0, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in US_CSR remains at 0. Otherwise, the receiver loads a 16-bit counter with the value programmed in TO. This counter is decremented at each bit period and reloaded each time a new character is received. If the counter reaches 0, the TIMEOUT bit in the Status Register rises.

The user can either:

- Obtain an interrupt when a time-out is detected after having received at least one character. This is performed by writing the Control Register (US_CR) with the STTTO (Start Time-out) bit at 1.
- Obtain a periodic interrupt while no character is received. This is performed by writing US_CR with the RETTO (Reload and Start Time-out) bit at 1.

If STTTO is performed, the counter clock is stopped until a first character is received. The idle state on RXD before the start of the frame does not provide a time-out. This prevents having to obtain a periodic interrupt and enables a wait of the end of frame when the idle state on RXD is detected.

If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

Figure 31-13 shows the block diagram of the Receiver Time-out feature.

Figure 31-13. Receiver Time-out Block Diagram

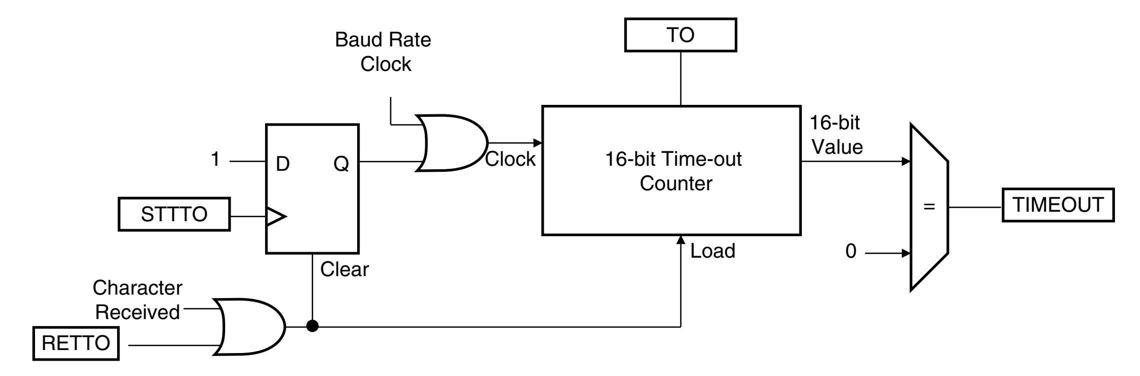


Table 31-8 gives the maximum time-out period for some standard baud rates.

Table 31-8. Maximum Time-out Period

Baud Rate	Bit Time	Time-out
bit/sec	µs	ms
600	1 667	109 225
1 200	833	54 613
2 400	417	27 306
4 800	208	13 653
9 600	104	6 827
14400	69	4 551
19200	52	3 413
28800	35	2 276
33400	30	1 962
56000	18	1 170
57600	17	1 138
200000	5	328

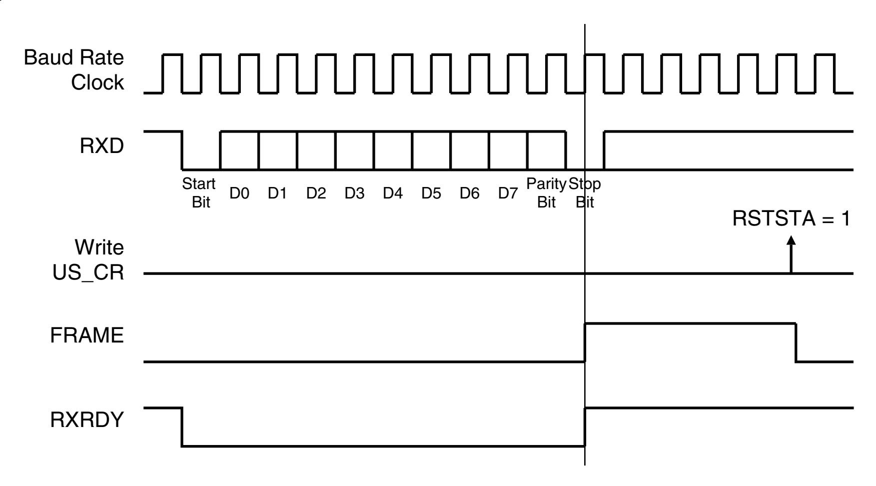


31.6.3.9 Framing Error

The receiver is capable of detecting framing errors. A framing error happens when the stop bit of a received character is detected at level 0. This can occur if the receiver and the transmitter are fully desynchronized.

A framing error is reported on the FRAME bit of the Channel Status Register (US_CSR). The FRAME bit is asserted in the middle of the stop bit as soon as the framing error is detected. It is cleared by writing the Control Register (US_CR) with the RSTSTA bit at 1.

Figure 31-14. Framing Error Status



31.6.3.10 Transmit Break

The user can request the transmitter to generate a break condition on the TXD line. A break condition drives the TXD line low during at least one complete character. It appears the same as a 0x00 character sent with the parity and the stop bits at 0. However, the transmitter holds the TXD line at least during one character until the user requests the break condition to be removed.

A break is transmitted by writing the Control Register (US_CR) with the STTBRK bit at 1. This can be performed at any time, either while the transmitter is empty (no character in either the Shift Register or in US_THR) or when a character is being transmitted. If a break is requested while a character is being shifted out, the character is first completed before the TXD line is held low.

Once STTBRK command is requested further STTBRK commands are ignored until the end of the break is completed.

The break condition is removed by writing US_CR with the STPBRK bit at 1. If the STPBRK is requested before the end of the minimum break duration (one character, including start, data, parity and stop bits), the transmitter ensures that the break condition completes.

The transmitter considers the break as though it is a character, i.e. the STTBRK and STPBRK commands are taken into account only if the TXRDY bit in US_CSR is at 1 and the start of the break condition clears the TXRDY and TXEMPTY bits as if a character is processed.

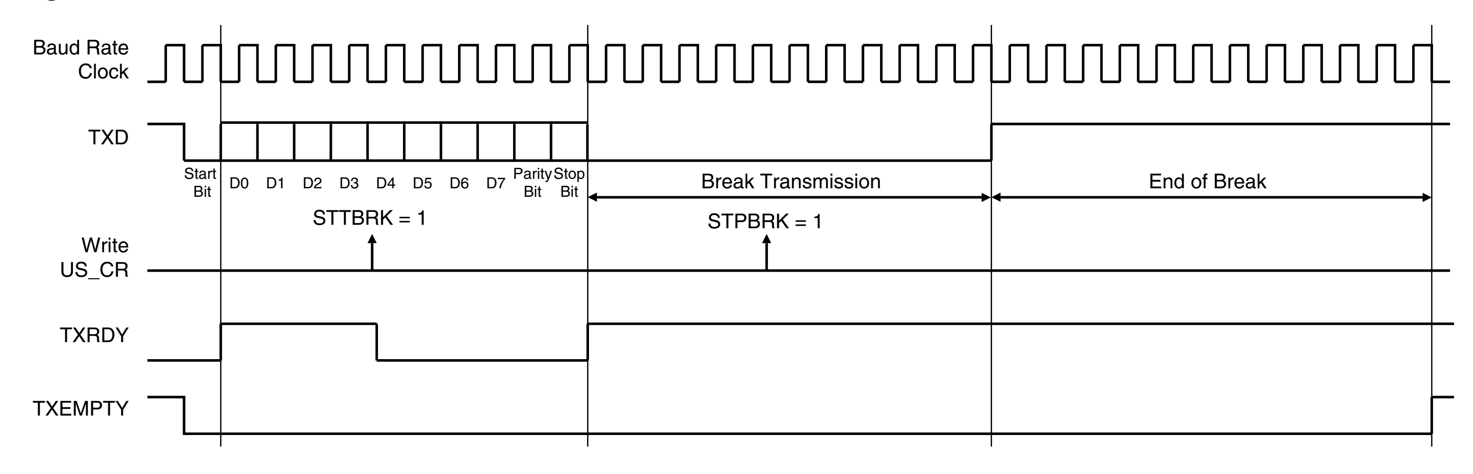
Writing US_CR with the both STTBRK and STPBRK bits at 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding Register while a break is pending, but not started, is ignored.

After the break condition, the transmitter returns the TXD line to 1 for a minimum of 12 bit times. Thus, the transmitter ensures that the remote receiver detects correctly the end of break and the start of the next character. If the timeguard is programmed with a value higher than 12, the TXD line is held high for the timeguard period.

After holding the TXD line for this period, the transmitter resumes normal operations.

Figure 31-15 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

Figure 31-15. Break Transmission



31.6.3.11 *Receive Break*

The receiver detects a break condition when all data, parity and stop bits are low. This corresponds to detecting a framing error with data at 0x00, but FRAME remains low.

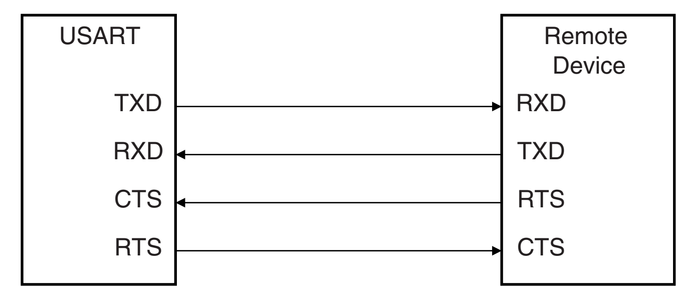
When the low stop bit is detected, the receiver asserts the RXBRK bit in US_CSR. This bit may be cleared by writing the Control Register (US_CR) with the bit RSTSTA at 1.

An end of receive break is detected by a high level for at least 2/16 of a bit period in asynchronous operating mode or one sample at high level in synchronous operating mode. The end of break detection also asserts the RXBRK bit.

31.6.3.12 *Hardware Handshaking*

The USART features a hardware handshaking out-of-band flow control. The RTS and CTS pins are used to connect with the remote device, as shown in Figure 31-16.

Figure 31-16. Connection with a Remote Device for Hardware Handshaking



Setting the USART to operate with hardware handshaking is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x2.

The USART behavior when hardware handshaking is enabled is the same as the behavior in standard synchronous or asynchronous mode, except that the receiver drives the RTS pin as described below and the level on the CTS pin modifies the behavior of the transmitter as described below. Using this mode requires using the PDC channel for reception. The transmitter can handle hardware handshaking in any case.

Figure 31-17 shows how the receiver operates if hardware handshaking is enabled. The RTS pin is driven high if the receiver is disabled and if the status RXBUFF (Receive Buffer Full) coming from the PDC channel is high. Normally, the remote device does not start transmitting while its CTS pin (driven by RTS) is high. As soon as the Receiver is enabled, the RTS falls, indicating to the remote device that it can start transmitting. Defining a new buffer to the PDC clears the status bit RXBUFF and, as a result, asserts the pin RTS low.

Figure 31-17. Receiver Behavior when Operating with Hardware Handshaking

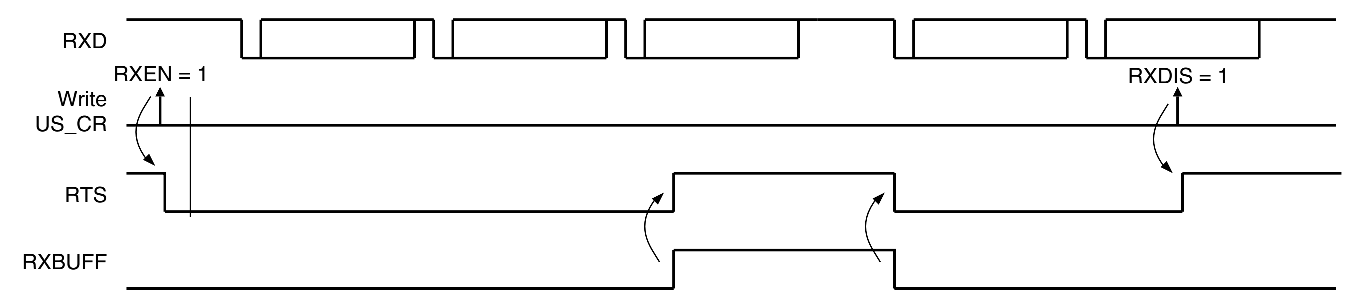
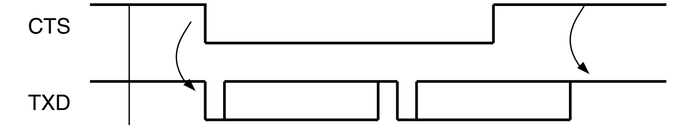


Figure 31-18 shows how the transmitter operates if hardware handshaking is enabled. The CTS pin disables the transmitter. If a character is being processing, the transmitter is disabled only after the completion of the current character and transmission of the next character happens as soon as the pin CTS falls.

Figure 31-18. Transmitter Behavior when Operating with Hardware Handshaking



31.6.4 ISO7816 Mode

The USART features an ISO7816-compatible operating mode. This mode permits interfacing with smart cards and Security Access Modules (SAM) communicating through an ISO7816 link. Both T = 0 and T = 1 protocols defined by the ISO7816 specification are supported.

Setting the USART in ISO7816 mode is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x4 for protocol T = 0 and to the value 0x5 for protocol T = 1.

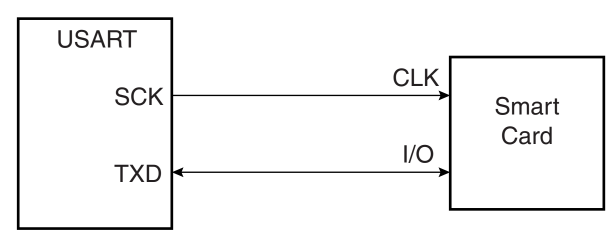
31.6.4.1 ISO7816 Mode Overview

The ISO7816 is a half duplex communication on only one bidirectional line. The baud rate is determined by a division of the clock provided to the remote device (see ”Baud Rate Generator” on page 383).

The USART connects to a smart card as shown in Figure 31-19. The TXD line becomes bidirectional and the Baud Rate Generator feeds the ISO7816 clock on the SCK pin. As the TXD pin becomes bidirectional, its output remains driven by the output of the transmitter but only

when the transmitter is active while its input is directed to the input of the receiver. The USART is considered as the master of the communication as it generates the clock.

Figure 31-19. Connection of a Smart Card to the USART



When operating in ISO7816, either in T = 0 or T = 1 modes, the character format is fixed. The configuration is 8 data bits, even parity and 1 or 2 stop bits, regardless of the values programmed in the CHRL, MODE9, PAR and CHMODE fields. MSBF can be used to transmit LSB or MSB first.

The USART cannot operate concurrently in both receiver and transmitter modes as the communication is unidirectional at a time. It has to be configured according to the required mode by enabling or disabling either the receiver or the transmitter as desired. Enabling both the receiver and the transmitter at the same time in ISO7816 mode may lead to unpredictable results.

The ISO7816 specification defines an inverse transmission format. Data bits of the character must be transmitted on the I/O line at their negative value. The USART does not support this format and the user has to perform an exclusive OR on the data before writing it in the Transmit Holding Register (US_THR) or after reading it in the Receive Holding Register (US_RHR).

31.6.4.2 Protocol T = 0

In T = 0 protocol, a character is made up of one start bit, eight data bits, one parity bit and one guard time, which lasts two bit times. The transmitter shifts out the bits and does not drive the I/O line during the guard time.

If no parity error is detected, the I/O line remains at 1 during the guard time and the transmitter can continue with the transmission of the next character, as shown in Figure 31-20.

If a parity error is detected by the receiver, it drives the I/O line at 0 during the guard time, as shown in Figure 31-21. This error bit is also named NACK, for Non Acknowledge. In this case, the character lasts 1 bit time more, as the guard time length is the same and is added to the error bit time which lasts 1 bit time.

When the USART is the receiver and it detects an error, it does not load the erroneous character in the Receive Holding Register (US_RHR). It appropriately sets the PARE bit in the Status Register (US_SR) so that the software can handle the error.

Figure 31-20. T = 0 Protocol without Parity Error

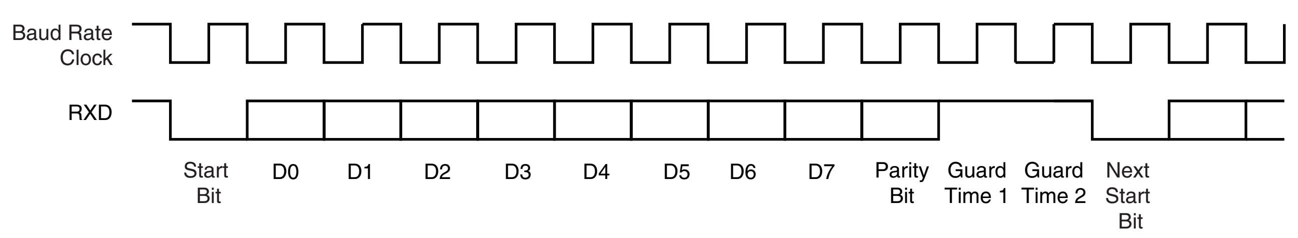
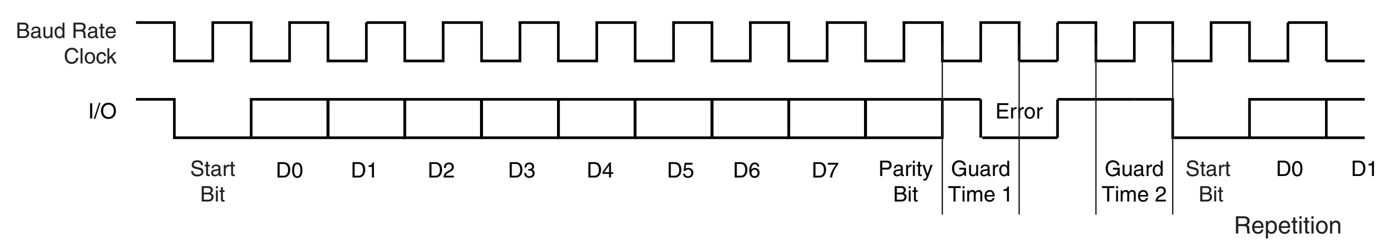


Figure 31-21. T = 0 Protocol with Parity Error



- Receive Error Counter

The USART receiver also records the total number of errors. This can be read in the Number of Error (US_NER) register. The NB_ERRORS field can record up to 255 errors. Reading US_NER automatically clears the NB_ERRORS field.

- Receive NACK Inhibit

The USART can also be configured to inhibit an error. This can be achieved by setting the INACK bit in the Mode Register (US_MR). If INACK is at 1, no error signal is driven on the I/O line even if a parity bit is detected, but the INACK bit is set in the Status Register (US_SR). The INACK bit can be cleared by writing the Control Register (US_CR) with the RSTNACK bit at 1.

Moreover, if INACK is set, the erroneous received character is stored in the Receive Holding Register, as if no error occurred. However, the RXRDY bit does not raise.

- Transmit Character Repetition

When the USART is transmitting a character and gets a NACK, it can automatically repeat the character before moving on to the next one. Repetition is enabled by writing the MAX_ITERATION field in the Mode Register (US_MR) at a value higher than 0. Each character can be transmitted up to eight times; the first transmission plus seven repetitions.

If MAX_ITERATION does not equal zero, the USART repeats the character as many times as the value loaded in MAX_ITERATION.

When the USART repetition number reaches MAX_ITERATION, the ITERATION bit is set in the Channel Status Register (US_CSR). If the repetition of the character is acknowledged by the receiver, the repetitions are stopped and the iteration counter is cleared.

The ITERATION bit in US_CSR can be cleared by writing the Control Register with the RSIT bit at 1.

- Disable Successive Receive NACK

The receiver can limit the number of successive NACKs sent back to the remote transmitter. This is programmed by setting the bit DSNACK in the Mode Register (US_MR). The maximum number of NACK transmitted is programmed in the MAX_ITERATION field. As soon as MAX_ITERATION is reached, the character is considered as correct, an acknowledge is sent on the line and the ITERATION bit in the Channel Status Register is set.

31.6.4.3 Protocol T = 1

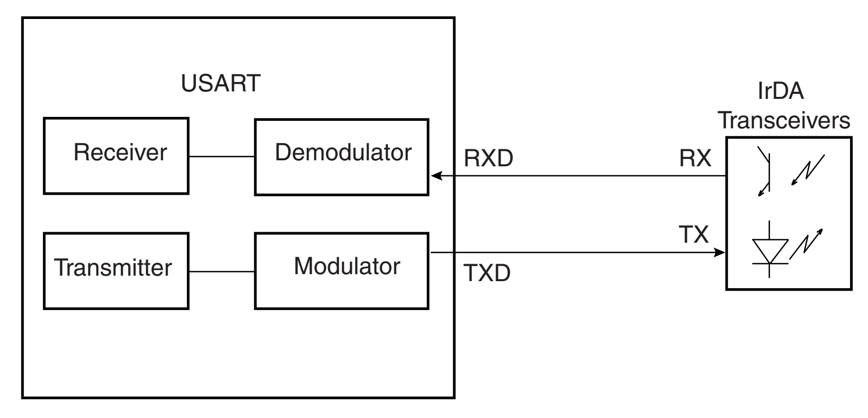
When operating in ISO7816 protocol T = 1, the transmission is similar to an asynchronous format with only one stop bit. The parity is generated when transmitting and checked when receiving. Parity error detection sets the PARE bit in the Channel Status Register (US_CSR).

31.6.5 IrDA Mode

The USART features an IrDA mode supplying half-duplex point-to-point wireless communication. It embeds the modulator and demodulator which allows a glueless connection to the infrared transceivers, as shown in Figure 31-22. The modulator and demodulator are compliant with the IrDA specification version 1.1 and support data transfer speeds ranging from 2.4 Kb/s to 115.2 Kb/s.

The USART IrDA mode is enabled by setting the USART_MODE field in the Mode Register (US_MR) to the value 0x8. The IrDA Filter Register (US_IF) allows configuring the demodulator filter. The USART transmitter and receiver operate in a normal asynchronous mode and all parameters are accessible. Note that the modulator and the demodulator are activated.

Figure 31-22. Connection to IrDA Transceivers



The receiver and the transmitter must be enabled or disabled according to the direction of the transmission to be managed.

31.6.5.1 IrDA Modulation

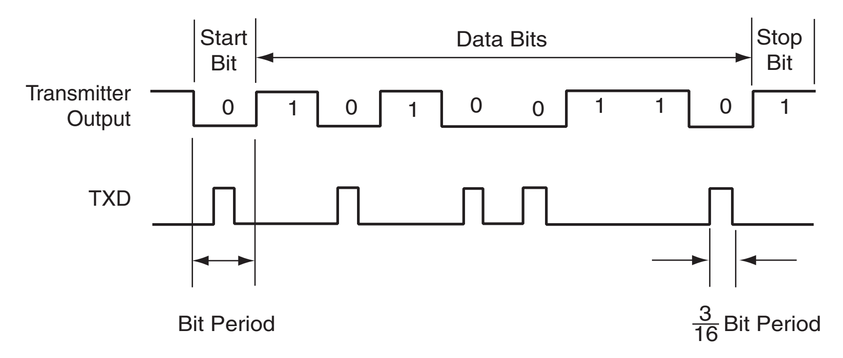
For baud rates up to and including 115.2 Kbits/sec, the RZI modulation scheme is used. “0” is represented by a light pulse of 3/16th of a bit time. Some examples of signal pulse duration are shown in Table 31-9.

Table 31-9. IrDA Pulse Duration

Baud Rate	Pulse Duration (3/16)
2.4 Kb/s	78.13 µs
9.6 Kb/s	19.53 µs
19.2 Kb/s	9.77 µs
38.4 Kb/s	4.88 µs
57.6 Kb/s	3.26 µs
115.2 Kb/s	1.63 µs

Figure 31-23 shows an example of character transmission.

Figure 31-23. IrDA Modulation



31.6.5.2 IrDA Baud Rate

Table 31-10 gives some examples of CD values, baud rate error and pulse duration. Note that the requirement on the maximum acceptable error of ±1.87% must be met.

Table 31-10. IrDA Baud Rate Error

Peripheral Clock	Baud Rate	CD	Baud Rate Error	Pulse Time
3 686 400	115 200	2	0.00%	1.63
20 000 000	115 200	11	1.38%	1.63
32 768 000	115 200	18	1.25%	1.63
40 000 000	115 200	22	1.38%	1.63
3 686 400	57 600	4	0.00%	3.26
20 000 000	57 600	22	1.38%	3.26
32 768 000	57 600	36	1.25%	3.26
40 000 000	57 600	43	0.93%	3.26
3 686 400	38 400	6	0.00%	4.88
20 000 000	38 400	33	1.38%	4.88
32 768 000	38 400	53	0.63%	4.88
40 000 000	38 400	65	0.16%	4.88
3 686 400	19 200	12	0.00%	9.77
20 000 000	19 200	65	0.16%	9.77
32 768 000	19 200	107	0.31%	9.77
40 000 000	19 200	130	0.16%	9.77
3 686 400	9 600	24	0.00%	19.53
20 000 000	9 600	130	0.16%	19.53
32 768 000	9 600	213	0.16%	19.53
40 000 000	9 600	260	0.16%	19.53
3 686 400	2 400	96	0.00%	78.13
20 000 000	2 400	521	0.03%	78.13
32 768 000	2 400	853	0.04%	78.13

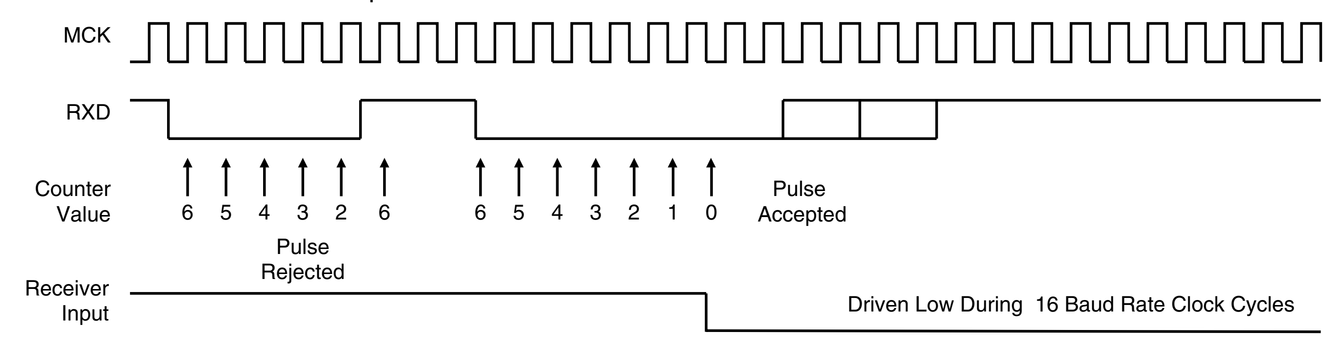


31.6.5.3 IrDA Demodulator

The demodulator is based on the IrDA Receive filter comprised of an 8-bit down counter which is loaded with the value programmed in US_IF. When a falling edge is detected on the RXD pin, the Filter Counter starts counting down at the Master Clock (MCK) speed. If a rising edge is detected on the RXD pin, the counter stops and is reloaded with US_IF. If no rising edge is detected when the counter reaches 0, the input of the receiver is driven low during one bit time.

Figure 31-24 illustrates the operations of the IrDA demodulator.

Figure 31-24. IrDA Demodulator Operations

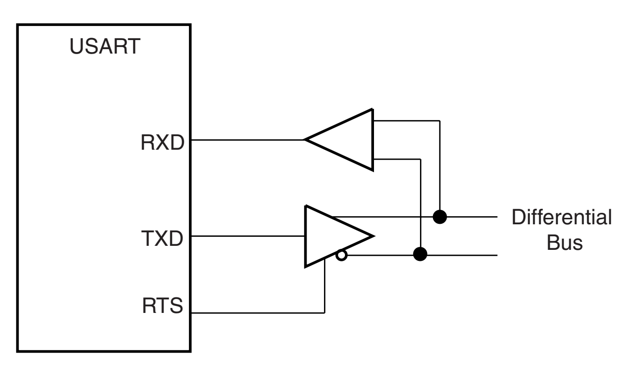


As the IrDA mode uses the same logic as the ISO7816, note that the FI_DI_RATIO field in US_FIDI must be set to a value higher than 0 in order to assure IrDA communications operate correctly.

31.6.6 RS485 Mode

The USART features the RS485 mode to enable line driver control. While operating in RS485 mode, the USART behaves as though in asynchronous or synchronous mode and configuration of all the parameters is possible. The difference is that the RTS pin is driven high when the transmitter is operating. The behavior of the RTS pin is controlled by the TXEMPTY bit. A typical connection of the USART to a RS485 bus is shown in Figure 31-25.

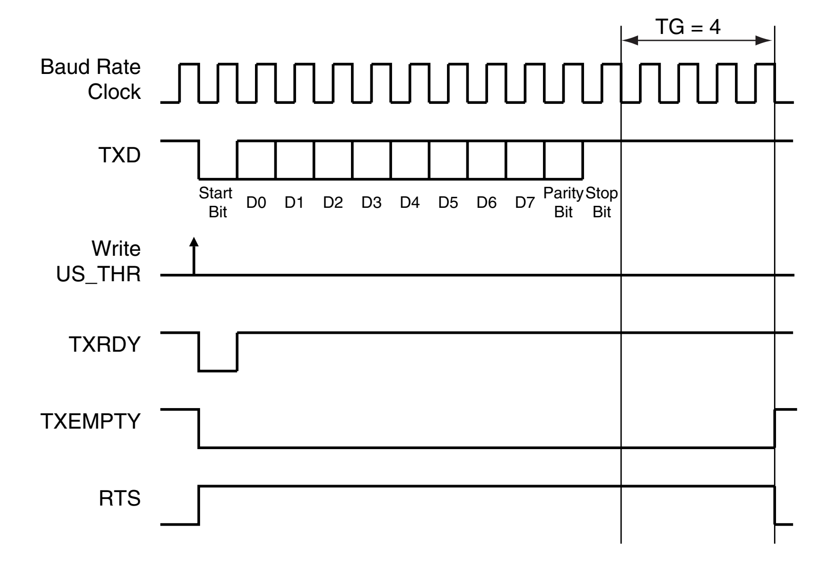
Figure 31-25. Typical Connection to a RS485 Bus



The USART is set in RS485 mode by programming the USART_MODE field in the Mode Register (US_MR) to the value 0x1.

The RTS pin is at a level inverse to the TXEMPTY bit. Significantly, the RTS pin remains high when a timeguard is programmed so that the line can remain driven after the last character completion. Figure 31-26 gives an example of the RTS waveform during a character transmission when the timeguard is enabled.

Figure 31-26. Example of RTS Drive with Timeguard



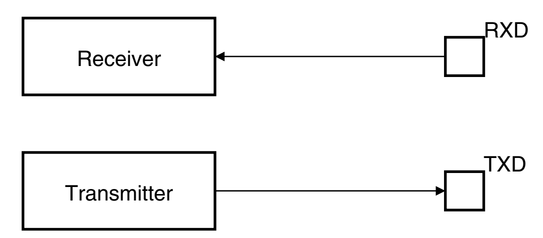
31.6.7 Test Modes

The USART can be programmed to operate in three different test modes. The internal loop-back capability allows on-board diagnostics. In the loopback mode the USART interface pins are disconnected or not and reconfigured for loopback internally or externally.

31.6.7.1 Normal Mode

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

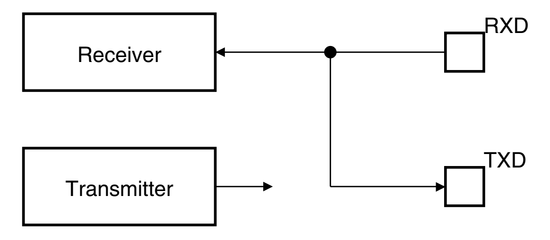
Figure 31-27. Normal Mode Configuration



31.6.7.2 Automatic Echo Mode

Automatic echo mode allows bit-by-bit retransmission. When a bit is received on the RXD pin, it is sent to the TXD pin, as shown in Figure 31-28. Programming the transmitter has no effect on the TXD pin. The RXD pin is still connected to the receiver input, thus the receiver remains active.

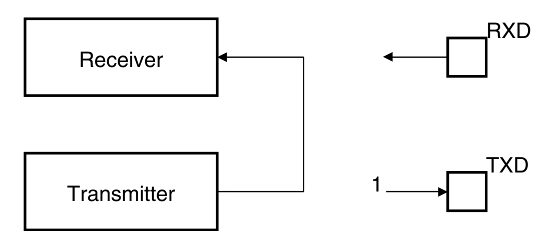
Figure 31-28. Automatic Echo Mode Configuration



31.6.7.3 Local Loopback Mode

Local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 31-29. The TXD and RXD pins are not used. The RXD pin has no effect on the receiver and the TXD pin is continuously driven high, as in idle state.

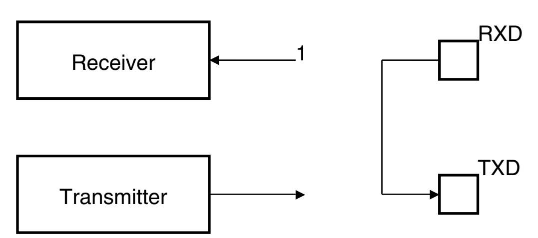
Figure 31-29. Local Loopback Mode Configuration



31.6.7.4 Remote Loopback Mode

Remote loopback mode directly connects the RXD pin to the TXD pin, as shown in Figure 31-30. The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

Figure 31-30. Remote Loopback Mode Configuration



31.7 USART User Interface

Table 31-11. USART Memory Map

Offset	Register	Name	Access	Reset State
0x0000	Control Register	US_CR	Write-only	–
0x0004	Mode Register	US_MR	Read/Write	–
0x0008	Interrupt Enable Register	US_IER	Write-only	–
0x000C	Interrupt Disable Register	US_IDR	Write-only	–
0x0010	Interrupt Mask Register	US_IMR	Read-only	0x0
0x0014	Channel Status Register	US_CSR	Read-only	–
0x0018	Receiver Holding Register	US_RHR	Read-only	0x0
0x001C	Transmitter Holding Register	US_THR	Write-only	–
0x0020	Baud Rate Generator Register	US_BRGR	Read/Write	0x0
0x0024	Receiver Time-out Register	US_RTOR	Read/Write	0x0
0x0028	Transmitter Timeguard Register	US_TTGR	Read/Write	0x0
0x2C - 0x3C	Reserved	–	–	–
0x0040	FI DI Ratio Register	US_FIDI	Read/Write	0x174
0x0044	Number of Errors Register	US_NER	Read-only	–
0x0048	Reserved	–	–	–
0x004C	IrDA Filter Register	US_IF	Read/Write	0x0
0x5C - 0xFC	Reserved	–	–	–
0x100 - 0x128	Reserved for PDC Registers	–	–	–

31.7.1 USART Control Register

Name: US_CR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	RTSDIS	RTSEN	–	–

15	14	13	12	11	10	9	8
RETTO	RSTNACK	RSTIT	SENDA	STTTO	STPBRK	STTBRK	RSTSTA

7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

- **RSTRX: Reset Receiver**

0: No effect.

1: Resets the receiver.

- **RSTTX: Reset Transmitter**

0: No effect.

1: Resets the transmitter.

- **RXEN: Receiver Enable**

0: No effect.

1: Enables the receiver, if RXDIS is 0.

- **RXDIS: Receiver Disable**

0: No effect.

1: Disables the receiver.

- **TXEN: Transmitter Enable**

0: No effect.

1: Enables the transmitter if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0: No effect.

1: Disables the transmitter.

- **RSTSTA: Reset Status Bits**

0: No effect.

1: Resets the status bits PARE, FRAME, OVRE and RXBRK in the US_CSR.

- **STTBRK: Start Break**

0: No effect.

1: Starts transmission of a break after the characters present in US_THR and the Transmit Shift Register have been transmitted. No effect if a break is already being transmitted.

- **STPBRK: Stop Break**

0: No effect.

1: Stops transmission of the break after a minimum of one character length and transmits a high level during 12-bit periods. No effect if no break is being transmitted.

- **STTTO: Start Time-out**

0: No effect

1: Starts waiting for a character before clocking the time-out counter.

- **SENDA: Send Address**

0: No effect.

1: In Multidrop Mode only, the next character written to the US_THR is sent with the address bit set.

- **RSTIT: Reset Iterations**

0: No effect.

1: Resets ITERATION in US_CSR. No effect if the ISO7816 is not enabled.

- **RSTNACK: Reset Non Acknowledge**

0: No effect

1: Resets NACK in US_CSR.

- **RETTO: Rearm Time-out**

0: No effect

1: Restart Time-out

- **RTSEN: Request to Send Enable**

0: No effect.

1: Drives the pin RTS to 0.

- **RTSDIS: Request to Send Disable**

0: No effect.

1: Drives the pin RTS to 1.

31.7.2 USART Mode Register

Name: US_MR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	FILTER	–	MAX_ITERATION		

23	22	21	20	19	18	17	16
–	–	DSNACK	INACK	OVER	CLKO	MODE9	MSBF

15	14	13	12	11	10	9	8
CHMODE		NBSTOP		PAR			SYNC

7	6	5	4	3	2	1	0
CHRL		USCLKS		USART_MODE			

- **USART_MODE**

USART_MODE				Mode of the USART
0	0	0	0	Normal
0	0	0	1	RS485
0	0	1	0	Hardware Handshaking
0	0	1	1	Reserved
0	1	0	0	IS07816 Protocol: T = 0
0	1	0	1	Reserved
0	1	1	0	IS07816 Protocol: T = 1
0	1	1	1	Reserved
1	0	0	0	IrDA
1	1	x	x	Reserved

- **USCLKS: Clock Selection**

USCLKS		Selected Clock
0	0	MCK
0	1	MCK / DIV
1	0	Reserved
1	1	SCK

- **CHRL: Character Length.**

CHRL		Character Length
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	1	8 bits

- **SYNC: Synchronous Mode Select**

0: USART operates in Asynchronous Mode.

1: USART operates in Synchronous Mode.

- **PAR: Parity Type**

PAR			Parity Type
0	0	0	Even parity
0	0	1	Odd parity
0	1	0	Parity forced to 0 (Space)
0	1	1	Parity forced to 1 (Mark)
1	0	x	No parity
1	1	x	Multidrop mode

- **NBSTOP: Number of Stop Bits**

NBSTOP		Asynchronous (SYNC = 0)	Synchronous (SYNC = 1)
0	0	1 stop bit	1 stop bit
0	1	1.5 stop bits	Reserved
1	0	2 stop bits	2 stop bits
1	1	Reserved	Reserved

- **CHMODE: Channel Mode**

CHMODE		Mode Description
0	0	Normal Mode
0	1	Automatic Echo. Receiver input is connected to the TXD pin.
1	0	Local Loopback. Transmitter output is connected to the Receiver Input..
1	1	Remote Loopback. RXD pin is internally connected to the TXD pin.

- **MSBF: Bit Order**

0: Least Significant Bit is sent/received first.

1: Most Significant Bit is sent/received first.

- **MODE9: 9-bit Character Length**

0: CHRL defines character length.

1: 9-bit character length.

- **CKLO: Clock Output Select**

0: The USART does not drive the SCK pin.

1: The USART drives the SCK pin if USCLKS does not select the external clock SCK.

- **OVER: Oversampling Mode**

0: 16x Oversampling.

1: 8x Oversampling.

- **INACK: Inhibit Non Acknowledge**

0: The NACK is generated.

1: The NACK is not generated.

- **DSNACK: Disable Successive NACK**

0: NACK is sent on the ISO line as soon as a parity error occurs in the received character (unless INACK is set).

1: Successive parity errors are counted up to the value specified in the MAX_ITERATION field. These parity errors generate a NACK on the ISO line. As soon as this value is reached, no additional NACK is sent on the ISO line. The flag ITERATION is asserted.

- **MAX_ITERATION**

Defines the maximum number of iterations in mode ISO7816, protocol T= 0.

- **FILTER: Infrared Receive Line Filter**

0: The USART does not filter the receive line.

1: The USART filters the receive line using a three-sample filter (1/16-bit clock) (2 over 3 majority).

31.7.3 USART Interrupt Enable Register

Name: US_IER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	MANE	CTSIC	–	–	–

15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY Interrupt Enable**
- **TXRDY: TXRDY Interrupt Enable**
- **RXBRK: Receiver Break Interrupt Enable**
- **ENDRX: End of Receive Transfer Interrupt Enable**
- **ENDTX: End of Transmit Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **FRAME: Framing Error Interrupt Enable**
- **PARE: Parity Error Interrupt Enable**
- **TIMEOUT: Time-out Interrupt Enable**
- **TXEMPTY: TXEMPTY Interrupt Enable**
- **ITERATION: Iteration Interrupt Enable**
- **TXBUFE: Buffer Empty Interrupt Enable**
- **RXBUFF: Buffer Full Interrupt Enable**
- **NACK: Non Acknowledge Interrupt Enable**
- **CTSIC: Clear to Send Input Change Interrupt Enable**
- **MANE: Manchester Error Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.



31.7.4 USART Interrupt Disable Register

Name: US_IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	–	–	–

15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY Interrupt Disable**
- **TXRDY: TXRDY Interrupt Disable**
- **RXBRK: Receiver Break Interrupt Disable**
- **ENDRX: End of Receive Transfer Interrupt Disable**
- **ENDTX: End of Transmit Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **FRAME: Framing Error Interrupt Disable**
- **PARE: Parity Error Interrupt Disable**
- **TIMEOUT: Time-out Interrupt Disable**
- **TXEMPTY: TXEMPTY Interrupt Disable**
- **ITERATION: Iteration Interrupt Disable**
- **TXBUFE: Buffer Empty Interrupt Disable**
- **RXBUFF: Buffer Full Interrupt Disable**
- **NACK: Non Acknowledge Interrupt Disable**
- **CTSIC: Clear to Send Input Change Interrupt Disable**

31.7.5 USART Interrupt Mask Register

Name: US_IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	CTSIC	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY Interrupt Mask**
- **TXRDY: TXRDY Interrupt Mask**
- **RXBRK: Receiver Break Interrupt Mask**
- **ENDRX: End of Receive Transfer Interrupt Mask**
- **ENDTX: End of Transmit Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **FRAME: Framing Error Interrupt Mask**
- **PARE: Parity Error Interrupt Mask**
- **TIMEOUT: Time-out Interrupt Mask**
- **TXEMPTY: TXEMPTY Interrupt Mask**
- **ITERATION: Iteration Interrupt Mask**
- **TXBUFE: Buffer Empty Interrupt Mask**
- **RXBUFF: Buffer Full Interrupt Mask**
- **NACK: Non Acknowledge Interrupt Mask**
- **CTSIC: Clear to Send Input Change Interrupt Mask**

31.7.6 USART Channel Status Register

Name: US_CSR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
CTS	–	–	–	CTSIC	–	–	–

15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: Receiver Ready**

0: No complete character has been received since the last read of US_RHR or the receiver is disabled. If characters were being received when the receiver was disabled, RXRDY changes to 1 when the receiver is enabled.

1: At least one complete character has been received and US_RHR has not yet been read.

- **TXRDY: Transmitter Ready**

0: A character is in the US_THR waiting to be transferred to the Transmit Shift Register, or an STTBRK command has been requested, or the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.

1: There is no character in the US_THR.

- **RXBRK: Break Received/End of Break**

0: No Break received or End of Break detected since the last RSTSTA.

1: Break Received or End of Break detected since the last RSTSTA.

- **ENDRX: End of Receiver Transfer**

0: The End of Transfer signal from the Receive PDC channel is inactive.

1: The End of Transfer signal from the Receive PDC channel is active.

- **ENDTX: End of Transmitter Transfer**

0: The End of Transfer signal from the Transmit PDC channel is inactive.

1: The End of Transfer signal from the Transmit PDC channel is active.

- **OVRE: Overrun Error**

0: No overrun error has occurred since the last RSTSTA.

1: At least one overrun error has occurred since the last RSTSTA.

- **FRAME: Framing Error**

0: No stop bit has been detected low since the last RSTSTA.

1: At least one stop bit has been detected low since the last RSTSTA.

- **PARE: Parity Error**

0: No parity error has been detected since the last RSTSTA.

1: At least one parity error has been detected since the last RSTSTA.

- **TIMEOUT: Receiver Time-out**

0: There has not been a time-out since the last Start Time-out command or the Time-out Register is 0.



1: There has been a time-out since the last Start Time-out command.

- **TXEMPTY: Transmitter Empty**

0: There are characters in either US_THR or the Transmit Shift Register, or the transmitter is disabled.

1: There is at least one character in either US_THR or the Transmit Shift Register.

- **ITERATION: Max number of Repetitions Reached**

0: Maximum number of repetitions has not been reached since the last RSIT.

1: Maximum number of repetitions has been reached since the last RSIT.

- **TXBUFE: Transmission Buffer Empty**

0: The signal Buffer Empty from the Transmit PDC channel is inactive.

1: The signal Buffer Empty from the Transmit PDC channel is active.

- **RXBUFF: Reception Buffer Full**

0: The signal Buffer Full from the Receive PDC channel is inactive.

1: The signal Buffer Full from the Receive PDC channel is active.

- **NACK: Non Acknowledge**

0: No Non Acknowledge has not been detected since the last RSTNACK.

1: At least one Non Acknowledge has been detected since the last RSTNACK.

- **CTSIC: Clear to Send Input Change Flag**

0: No input change has been detected on the CTS pin since the last read of US_CSR.

1: At least one input change has been detected on the CTS pin since the last read of US_CSR.

- **CTS: Image of CTS Input**

0: CTS is at 0.

1: CTS is at 1.

31.7.7 USART Receive Holding Register

Name: US_RHR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
RXSYNH	–	–	–	–	–	–	RXCHR

7	6	5	4	3	2	1	0
RXCHR							

- **RXCHR: Received Character**

Last character received if RXRDY is set.

- **RXSYNH: Received Sync**

0: Last Character received is a Data.

1: Last Character received is a Command.

31.7.8 USART Transmit Holding Register

Name: US_THR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
TXSYNH	–	–	–	–	–	–	TXCHR

7	6	5	4	3	2	1	0
TXCHR							

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set.

- **TXSYNH: Sync Field to be transmitted**

0: The next character sent is encoded as a data. Start Frame Delimiter is DATA SYNC.

1: The next character sent is encoded as a command. Start Frame Delimiter is COMMAND SYNC.



31.7.9 USART Baud Rate Generator Register

Name: US_BRGR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–		

15	14	13	12	11	10	9	8
CD							

7	6	5	4	3	2	1	0
CD							

- **CD: Clock Divider**

CD	USART_MODE ≠ ISO7816			USART_MODE = ISO7816
	SYNC = 0		SYNC = 1	
	OVER = 0	OVER = 1		
0	Baud Rate Clock Disabled			
1 to 65535	Baud Rate = Selected Clock/16/CD	Baud Rate = Selected Clock/8/CD	Baud Rate = Selected Clock /CD	Baud Rate = Selected Clock/CD/FI_DI_RATIO

31.7.10 USART Receiver Time-out Register

Name: US_RTOR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
TO							

7	6	5	4	3	2	1	0
TO							

- **TO: Time-out Value**

0: The Receiver Time-out is disabled.

1 - 65535: The Receiver Time-out is enabled and the Time-out delay is TO x Bit Period.



31.7.11 USART Transmitter Timeguard Register

Name: US_TTGR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
TG							

- **TG: Timeguard Value**

0: The Transmitter Timeguard is disabled.

1 - 255: The Transmitter timeguard is enabled and the timeguard delay is TG x Bit Period.



31.7.12 USART FI DI RATIO Register

Name: US_FIDI

Access Type: Read/Write

Reset Value : 0x174

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	FI_DI_RATIO		

7	6	5	4	3	2	1	0
FI_DI_RATIO							

- **FI_DI_RATIO: FI Over DI Ratio Value**

0: If ISO7816 mode is selected, the Baud Rate Generator generates no signal.

1 - 2047: If ISO7816 mode is selected, the Baud Rate is the clock provided on SCK divided by FI_DI_RATIO.



31.7.13 USART Number of Errors Register

Name: US_NER

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
NB_ERRORS							

- **NB_ERRORS: Number of Errors**

Total number of errors that occurred during an ISO7816 transfer. This register automatically clears when read.

31.7.14 USART IrDA FILTER Register

Name: US_IF

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
IRDA_FILTER							

- **IRDA_FILTER: IrDA Filter**

Sets the filter of the IrDA demodulator.

32. Synchronous Serial Controller (SSC)

32.1 Overview

The Atmel Synchronous Serial Controller (SSC) provides a synchronous communication link with external devices. It supports many serial synchronous communication protocols generally used in audio and telecom applications such as I2S, Short Frame Sync, Long Frame Sync, etc.

The SSC contains an independent receiver and transmitter and a common clock divider. The receiver and the transmitter each interface with three signals: the TD/RD signal for data, the TK/RK signal for the clock and the TF/RF signal for the Frame Sync. The transfers can be programmed to start automatically or on different events detected on the Frame Sync signal.

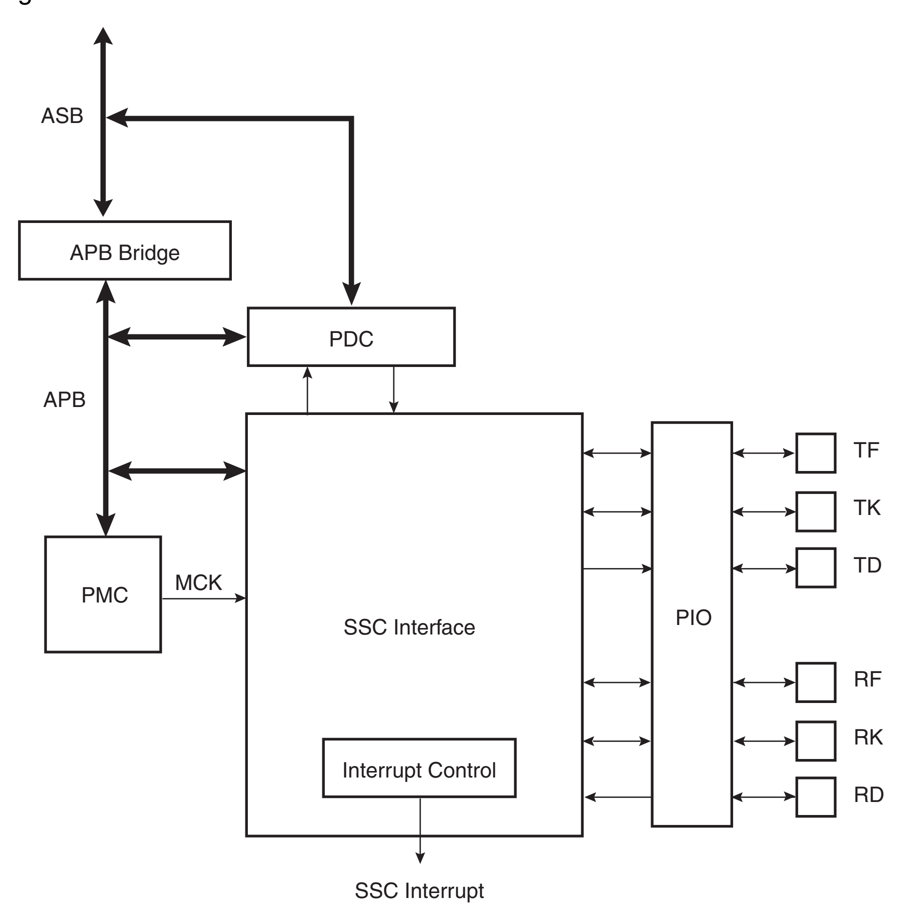
The SSC's high-level of programmability and its two dedicated PDC channels of up to 32 bits permit a continuous high bit rate data transfer without processor intervention.

Featuring connection to two PDC channels, the SSC permits interfacing with low processor overhead to the following:

- CODEC's in master or slave mode
- DAC through dedicated serial interface, particularly I2S
- Magnetic card reader

32.2 Block Diagram

Figure 32-1. Block Diagram



32.3 Application Block Diagram

Figure 32-2. Application Block Diagram

OS or RTOS Driver	Power Management	Interrupt Management	Test Management

SSC

Serial AUDIO	Codec	Time Slot Management	Frame Management	Line Interface

32.4 Pin Name List

Table 32-1. I/O Lines Description

Pin Name	Pin Description	Type
RF	Receiver Frame Synchro	Input/Output
RK	Receiver Clock	Input/Output
RD	Receiver Data	Input
TF	Transmitter Frame Synchro	Input/Output
TK	Transmitter Clock	Input/Output
TD	Transmitter Data	Output

32.5 Product Dependencies

32.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines.

Before using the SSC receiver, the PIO controller must be configured to dedicate the SSC receiver I/O lines to the SSC peripheral mode.

Before using the SSC transmitter, the PIO controller must be configured to dedicate the SSC transmitter I/O lines to the SSC peripheral mode.

32.5.2 Power Management

The SSC is not continuously clocked. The SSC interface may be clocked through the Power Management Controller (PMC), therefore the programmer must first configure the PMC to enable the SSC clock.

32.5.3 Interrupt

The SSC interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling interrupts requires programming the AIC before configuring the SSC.

All SSC interrupts can be enabled/disabled configuring the SSC Interrupt mask register. Each pending and unmasked SSC interrupt will assert the SSC interrupt line. The SSC interrupt service routine can get the interrupt origin by reading the SSC interrupt status register.

32.6 Functional Description

This chapter contains the functional description of the following: SSC Functional Block, Clock Management, Data format, Start, Transmitter, Receiver and Frame Sync.

The receiver and transmitter operate separately. However, they can work synchronously by programming the receiver to use the transmit clock and/or to start a data transfer when transmission starts. Alternatively, this can be done by programming the transmitter to use the receive clock and/or to start a data transfer when reception starts. The transmitter and the receiver can be programmed to operate with the clock signals provided on either the TK or RK pins. This allows the SSC to support many slave-mode data transfers. The maximum clock speed allowed on the TK and RK pins is the master clock divided by 2.

Figure 32-3. SSC Functional Block Diagram

32.6.1 Clock Management

The transmitter clock can be generated by:

- an external clock received on the TK I/O pad
- the receiver clock
- the internal clock divider

The receiver clock can be generated by:

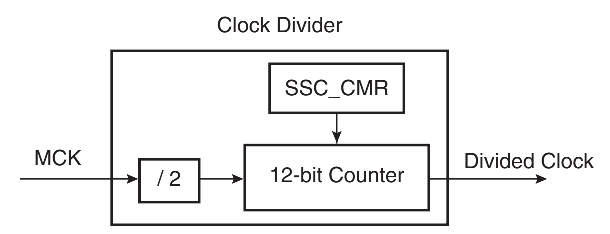
- an external clock received on the RK I/O pad
- the transmitter clock
- the internal clock divider

Furthermore, the transmitter block can generate an external clock on the TK I/O pad, and the receiver block can generate an external clock on the RK I/O pad.

This allows the SSC to support many Master and Slave Mode data transfers.

32.6.1.1 *Clock Divider*

Figure 32-4. Divided Clock Block Diagram



The Master Clock divider is determined by the 12-bit field DIV counter and comparator (so its maximal value is 4095) in the Clock Mode Register SSC_CMR, allowing a Master Clock division by up to 8190. The Divided Clock is provided to both the Receiver and Transmitter. When this field is programmed to 0, the Clock Divider is not used and remains inactive.

When DIV is set to a value equal to or greater than 1, the Divided Clock has a frequency of Master Clock divided by 2 times DIV. Each level of the Divided Clock has a duration of the Master Clock multiplied by DIV. This ensures a 50% duty cycle for the Divided Clock regardless of whether the DIV value is even or odd.

Figure 32-5. Divided Clock Generation

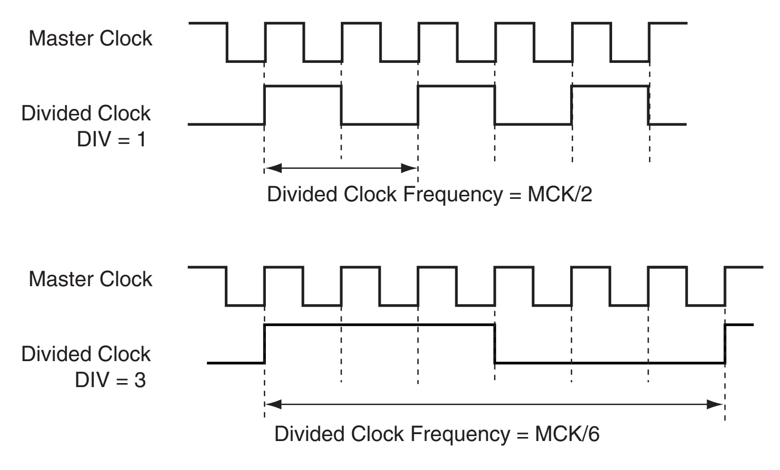


Table 32-2.

Maximum	Minimum
MCK / 2	MCK / 8190

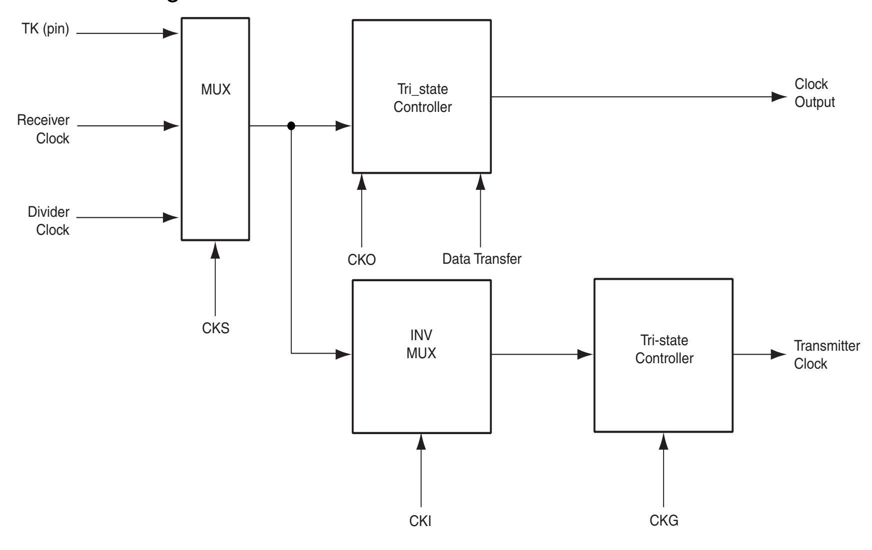
32.6.1.2 *Transmitter Clock Management*

The transmitter clock is generated from the receiver clock or the divider clock or an external clock scanned on the TK I/O pad. The transmitter clock is selected by the CKS field in SSC_TCMR (Transmit Clock Mode Register). Transmit Clock can be inverted independently by the CKI bits in SSC_TCMR.

The transmitter can also drive the TK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the SSC_TCMR register. The Transmit Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the TCMR register to select

TK pin (CKS field) and at the same time Continuous Transmit Clock (CKO field) might lead to unpredictable results.

Figure 32-6. Transmitter Clock Management

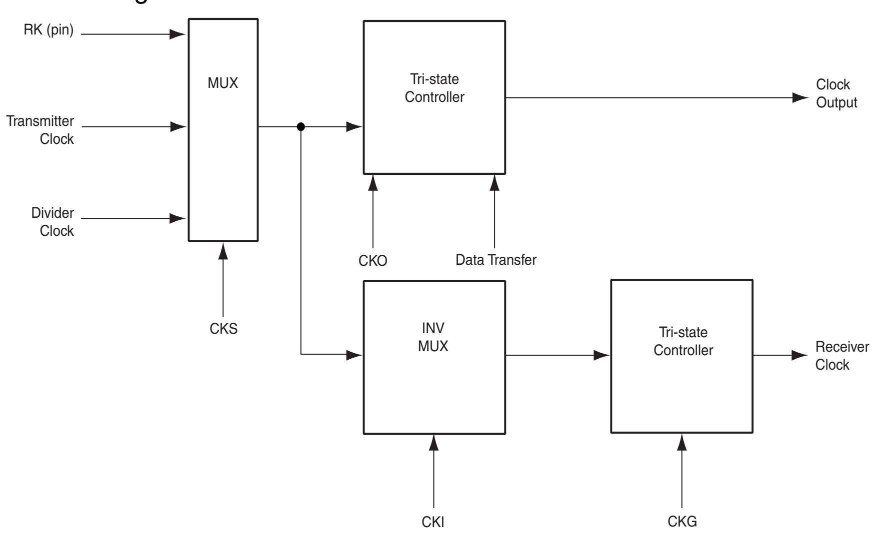


32.6.1.3 Receiver Clock Management

The receiver clock is generated from the transmitter clock or the divider clock or an external clock scanned on the RK I/O pad. The Receive Clock is selected by the CKS field in SSC_RCMR (Receive Clock Mode Register). Receive Clocks can be inverted independently by the CKI bits in SSC_RCMR.

The receiver can also drive the RK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the SSC_RCMR register. The Receive Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the RCMR register to select RK pin (CKS field) and at the same time Continuous Receive Clock (CKO field) can lead to unpredictable results.

Figure 32-7. Receiver Clock Management



32.6.1.4 *Serial Clock Ratio Considerations*

The Transmitter and the Receiver can be programmed to operate with the clock signals provided on either the TK or RK pins. This allows the SSC to support many slave-mode data transfers. In this case, the maximum clock speed allowed on the RK pin is:

– Master Clock divided by 2 if Receiver Frame Synchro is input

– Master Clock divided by 3 if Receiver Frame Synchro is output

In addition, the maximum clock speed allowed on the TK pin is:

– Master Clock divided by 6 if Transmit Frame Synchro is input

– Master Clock divided by 2 if Transmit Frame Synchro is output

32.6.2 Transmitter Operations

A transmitted frame is triggered by a start event and can be followed by synchronization data before data transmission.

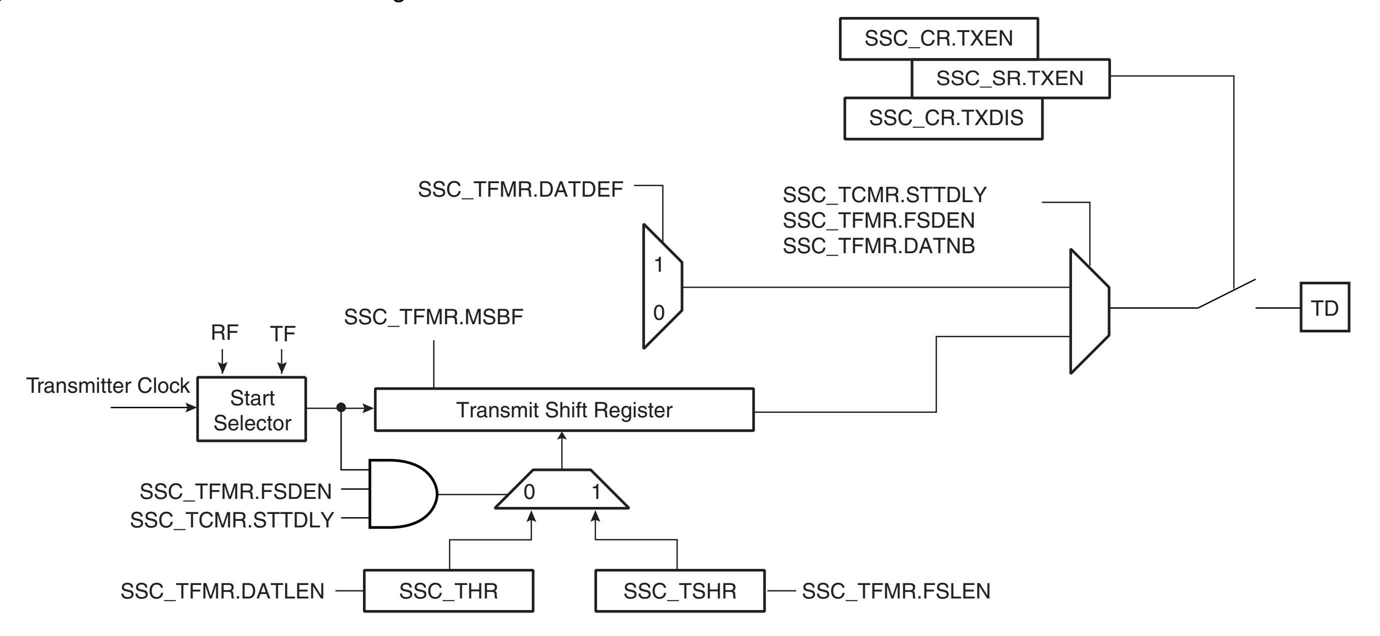
The start event is configured by setting the Transmit Clock Mode Register (SSC_TCMR). See “Start” on page 431.

The frame synchronization is configured setting the Transmit Frame Mode Register (SSC_TFMR). See “Frame Sync” on page 433.

To transmit data, the transmitter uses a shift register clocked by the transmitter clock signal and the start mode selected in the SSC_TCMR. Data is written by the application to the SSC_THR register then transferred to the shift register according to the data format selected.

When both the SSC_THR and the transmit shift register are empty, the status flag TXEMPTY is set in SSC_SR. When the Transmit Holding register is transferred in the Transmit shift register, the status flag TXRDY is set in SSC_SR and additional data can be loaded in the holding register.

Figure 32-8. Transmitter Block Diagram



32.6.3 Receiver Operations

A received frame is triggered by a start event and can be followed by synchronization data before data transmission.

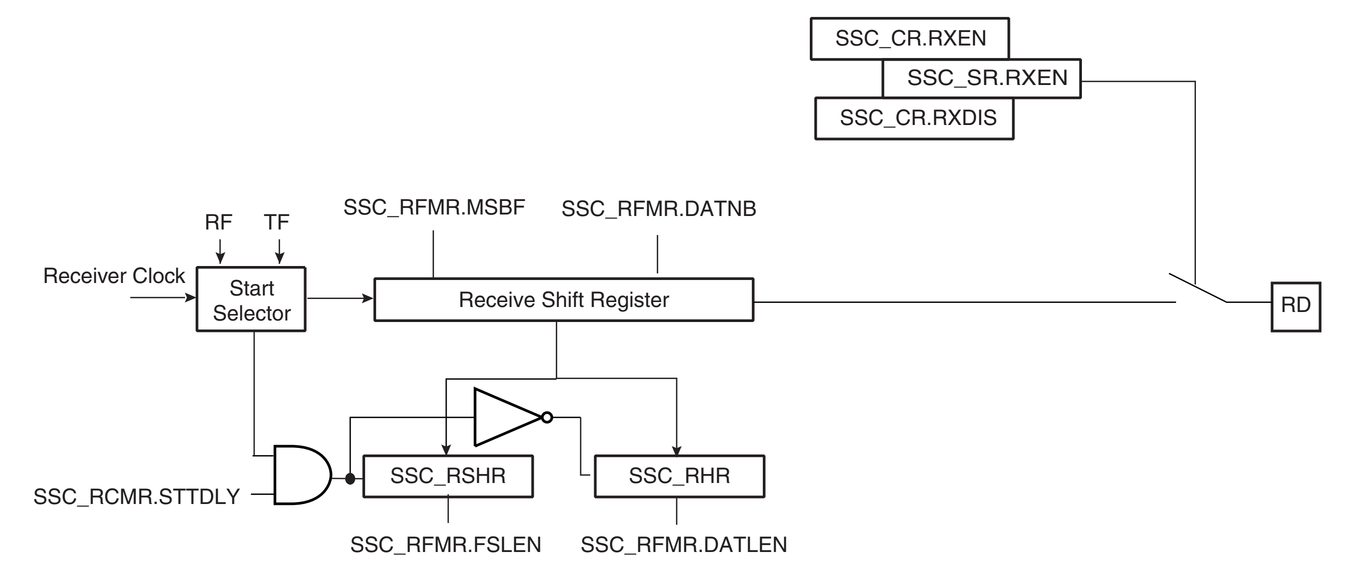
The start event is configured setting the Receive Clock Mode Register (SSC_RCMR). See "Start" on page 431.

The frame synchronization is configured setting the Receive Frame Mode Register (SSC_RFMR). See "Frame Sync" on page 433.

The receiver uses a shift register clocked by the receiver clock signal and the start mode selected in the SSC_RCMR. The data is transferred from the shift register depending on the data format selected.

When the receiver shift register is full, the SSC transfers this data in the holding register, the status flag RXRDY is set in SSC_SR and the data can be read in the receiver holding register. If another transfer occurs before read of the RHR register, the status flag OVERUN is set in SSC_SR and the receiver shift register is transferred in the RHR register.

Figure 32-9. Receiver Block Diagram



32.6.4 Start

The transmitter and receiver can both be programmed to start their operations when an event occurs, respectively in the Transmit Start Selection (START) field of SSC_TCMR and in the Receive Start Selection (START) field of SSC_RCMR.

Under the following conditions the start event is independently programmable:

- Continuous. In this case, the transmission starts as soon as a word is written in SSC_THR and the reception starts as soon as the Receiver is enabled.
- Synchronously with the transmitter/receiver
- On detection of a falling/rising edge on TF/RF
- On detection of a low level/high level on TF/RF
- On detection of a level change or an edge on TF/RF

A start can be programmed in the same manner on either side of the Transmit/Receive Clock Register (RCMR/TCMR). Thus, the start could be on TF (Transmit) or RF (Receive).

Moreover, the Receiver can start when data is detected in the bit stream with the Compare Functions.

Detection on TF/RF input/output is done by the field FSOS of the Transmit/Receive Frame Mode Register (TFMR/RFMR).

Figure 32-10. Transmit Start Mode

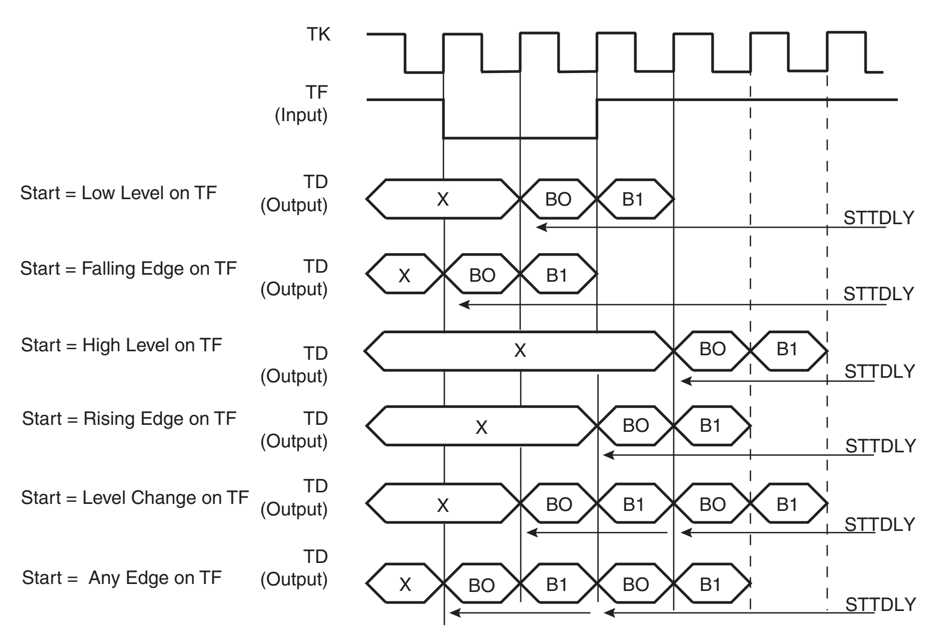
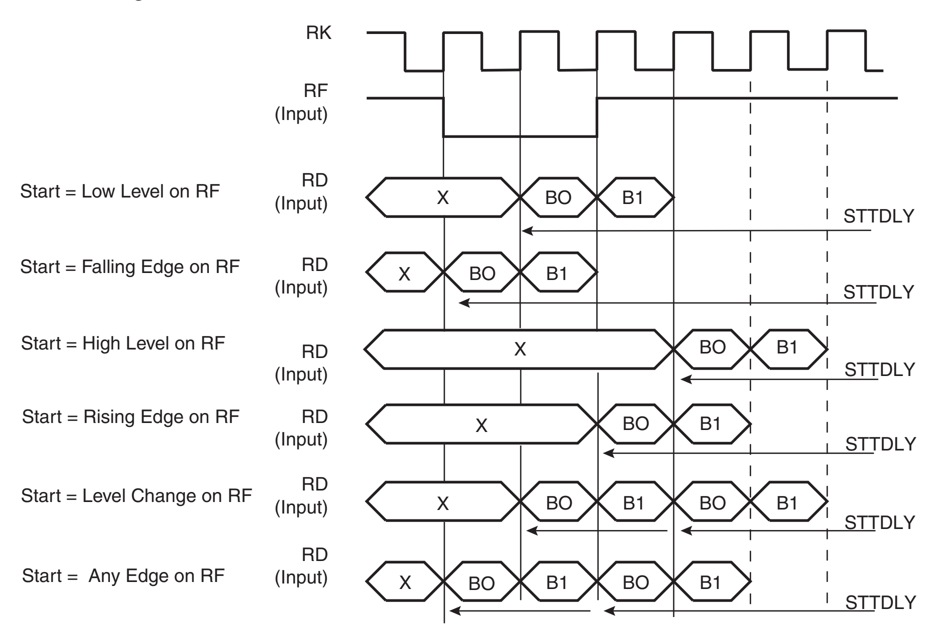


Figure 32-11. Receive Pulse/Edge Start Modes



32.6.5 Frame Sync

The Transmitter and Receiver Frame Sync pins, TF and RF, can be programmed to generate different kinds of frame synchronization signals. The Frame Sync Output Selection (FSOS) field in the Receive Frame Mode Register (SSC_RFMR) and in the Transmit Frame Mode Register (SSC_TFMR) are used to select the required waveform.

- Programmable low or high levels during data transfer are supported.
- Programmable high levels before the start of data transfers or toggling are also supported.

If a pulse waveform is selected, the Frame Sync Length (FSLEN) field in SSC_RFMR and SSC_TFMR programs the length of the pulse, from 1 bit time up to 16 bit time.

The periodicity of the Receive and Transmit Frame Sync pulse output can be programmed through the Period Divider Selection (PERIOD) field in SSC_RCMR and SSC_TCMR.

32.6.5.1 Frame Sync Data

Frame Sync Data transmits or receives a specific tag during the Frame Sync signal.

During the Frame Sync signal, the Receiver can sample the RD line and store the data in the Receive Sync Holding Register and the transmitter can transfer Transmit Sync Holding Register in the Shifter Register. The data length to be sampled/shifted out during the Frame Sync signal is programmed by the FSLEN field in SSC_RFMR/SSC_TFMR.

Concerning the Receive Frame Sync Data operation, if the Frame Sync Length is equal to or lower than the delay between the start event and the actual data reception, the data sampling operation is performed in the Receive Sync Holding Register through the Receive Shift Register.

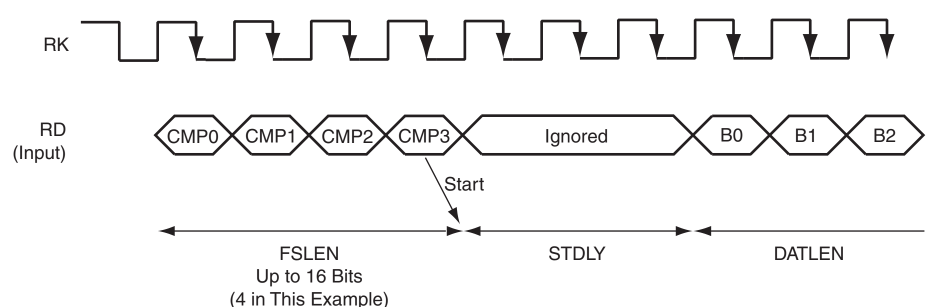
The Transmit Frame Sync Operation is performed by the transmitter only if the bit Frame Sync Data Enable (FSDEN) in SSC_TFMR is set. If the Frame Sync length is equal to or lower than the delay between the start event and the actual data transmission, the normal transmission has priority and the data contained in the Transmit Sync Holding Register is transferred in the Transmit Register, then shifted out.

32.6.5.2 Frame Sync Edge Detection

The Frame Sync Edge detection is programmed by the FSEDGE field in SSC_RFMR/SSC_TFMR. This sets the corresponding flags RXSYN/TXSYN in the SSC Status Register (SSC_SR) on frame synchro edge detection (signals RF/TF).

32.6.6 Receive Compare Modes

Figure 32-12. Receive Compare Modes





32.6.6.1 Compare Functions

Compare 0 can be one start event of the Receiver. In this case, the receiver compares at each new sample the last FSLEN bits received at the FSLEN lower bit of the data contained in the Compare 0 Register (SSC_RC0R). When this start event is selected, the user can program the Receiver to start a new data transfer either by writing a new Compare 0, or by receiving continuously until Compare 1 occurs. This selection is done with the bit (STOP) in SSC_RCMR.

32.6.7 Data Format

The data framing format of both the transmitter and the receiver are programmable through the Transmitter Frame Mode Register (SSC_TFMR) and the Receiver Frame Mode Register (SSC_RFMR). In either case, the user can independently select:

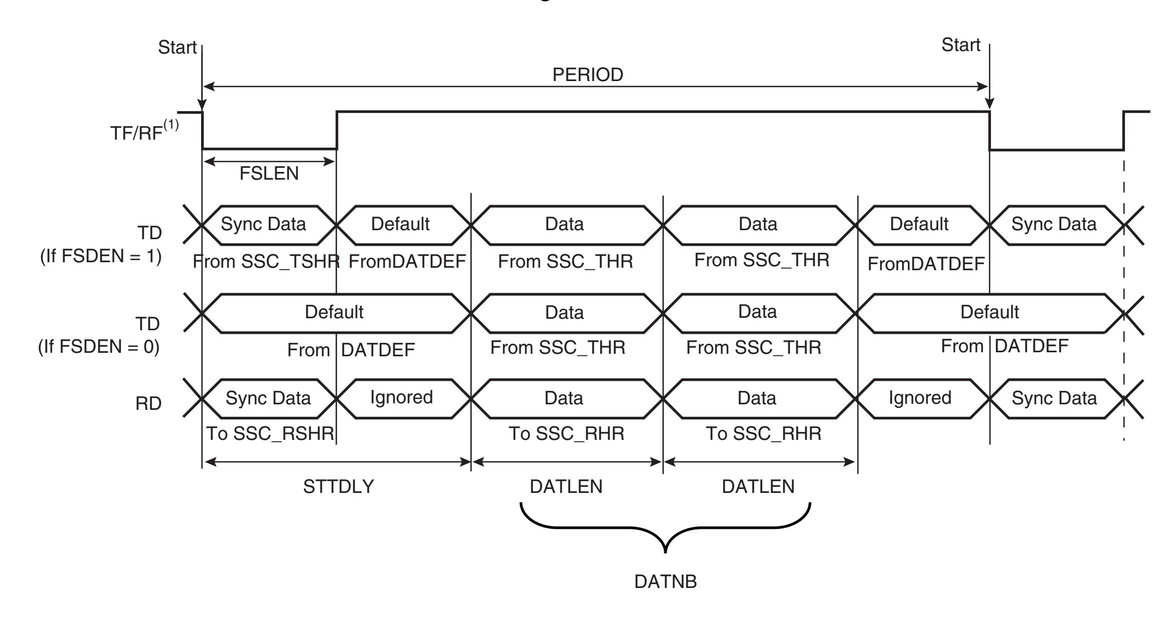
- the event that starts the data transfer (START)
- the delay in number of bit periods between the start event and the first data bit (STTDLY)
- the length of the data (DATLEN)
- the number of data to be transferred for each start event (DATNB).
- the length of synchronization transferred for each start event (FSLEN)
- the bit sense: most or lowest significant bit first (MSBF).

Additionally, the transmitter can be used to transfer synchronization and select the level driven on the TD pin while not in data transfer operation. This is done respectively by the Frame Sync Data Enable (FSDEN) and by the Data Default Value (DATDEF) bits in SSC_TFMR.

Table 32-3. Data Frame Registers

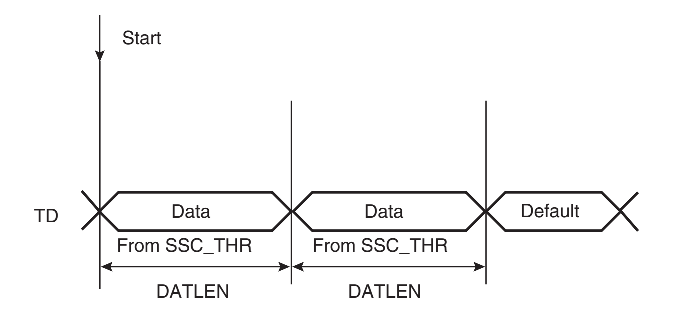
Transmitter	Receiver	Field	Length	Comment
SSC_TFMR	SSC_RFMR	DATLEN	Up to 32	Size of word
SSC_TFMR	SSC_RFMR	DATNB	Up to 16	Number of words transmitted in frame
SSC_TFMR	SSC_RFMR	MSBF		Most significant bit first
SSC_TFMR	SSC_RFMR	FSLEN	Up to 16	Size of Synchro data register
SSC_TFMR		DATDEF	0 or 1	Data default value ended
SSC_TFMR		FSDEN		Enable send SSC_TSHR
SSC_TCMR	SSC_RCMR	PERIOD	Up to 512	Frame size
SSC_TCMR	SSC_RCMR	STTDLY	Up to 255	Size of transmit start delay

Figure 32-13. Transmit and Receive Frame Format in Edge/Pulse Start Modes



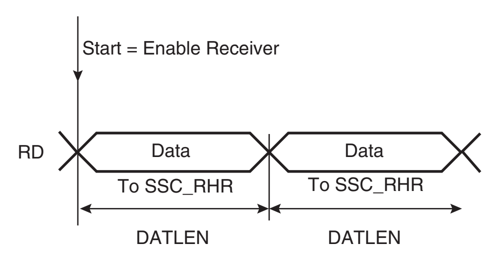
Note: 1. Example of input on falling edge of TF/RF.

Figure 32-14. Transmit Frame Format in Continuous Mode



Note: 1. STTDLY is set to 0. In this example, SSC_THR is loaded twice. FSDEN value has no effect on the transmission. SyncData cannot be output in continuous mode.

Figure 32-15. Receive Frame Format in Continuous Mode



Note: 1. STTDLY is set to 0.

32.6.8 Loop Mode

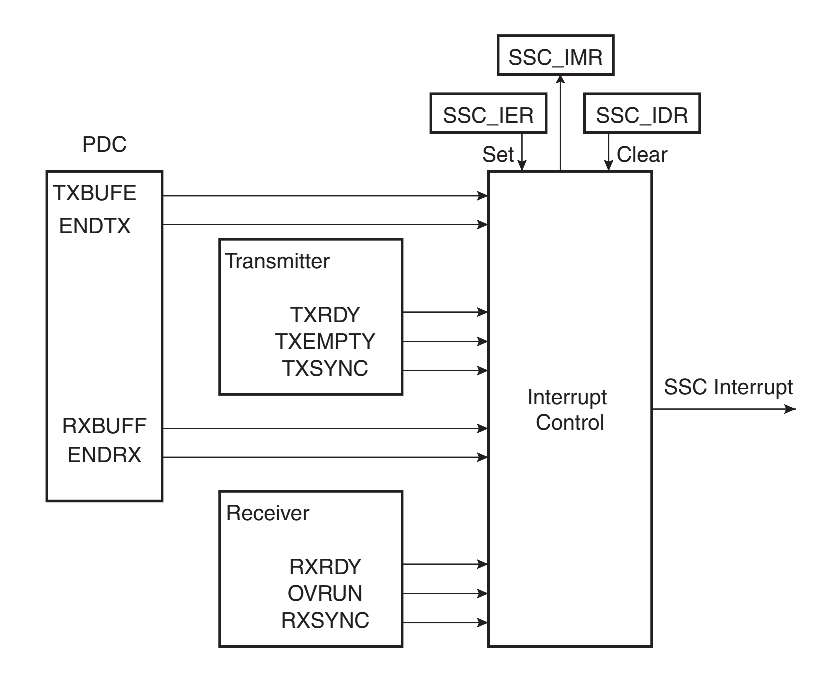
The receiver can be programmed to receive transmissions from the transmitter. This is done by setting the Loop Mode (LOOP) bit in SSC_RFMR. In this case, RD is connected to TD, RF is connected to TF and RK is connected to TK.

32.6.9 Interrupt

Most bits in SSC_SR have a corresponding bit in interrupt management registers.

The SSC can be programmed to generate an interrupt when it detects an event. The interrupt is controlled by writing SSC_IER (Interrupt Enable Register) and SSC_IDR (Interrupt Disable Register) These registers enable and disable, respectively, the corresponding interrupt by setting and clearing the corresponding bit in SSC_IMR (Interrupt Mask Register), which controls the generation of interrupts by asserting the SSC interrupt line connected to the AIC.

Figure 32-16. Interrupt Block Diagram



32.7 SSC Application Examples

The SSC can support several serial communication modes used in audio or high speed serial links. Some standard applications are shown in the following figures. All serial link applications supported by the SSC are not listed here.

Figure 32-17. Audio Application Block Diagram

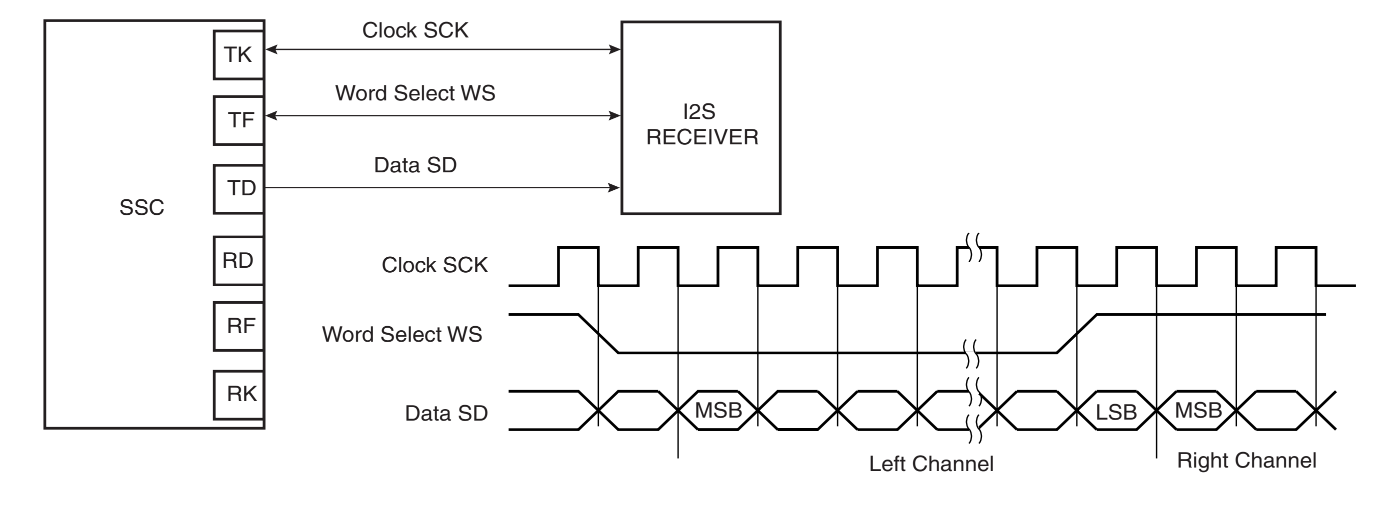


Figure 32-18. Codec Application Block Diagram

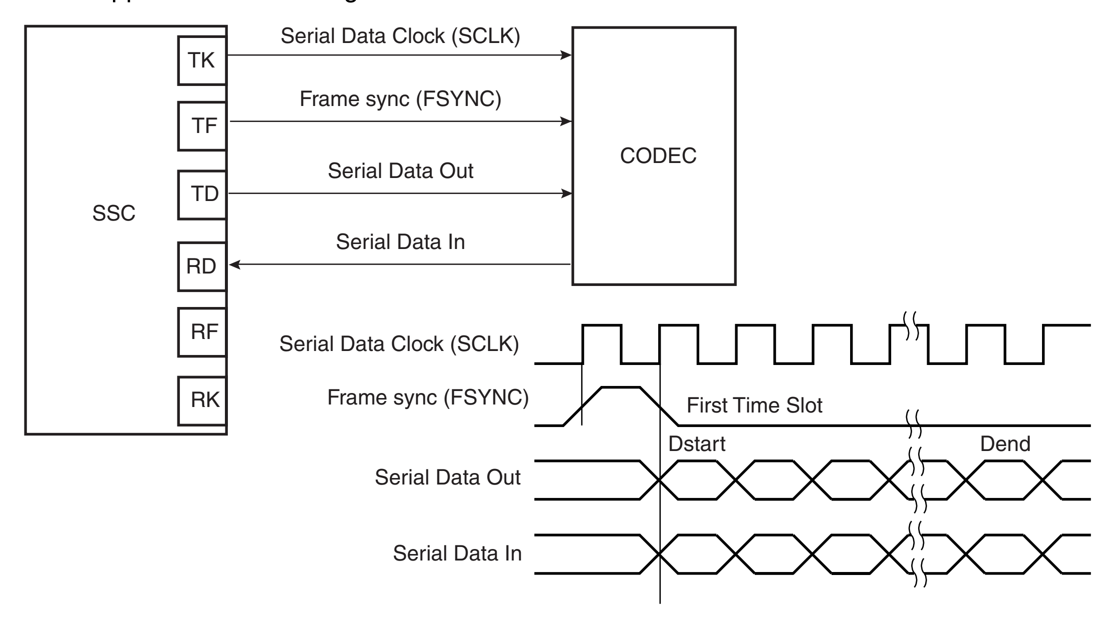
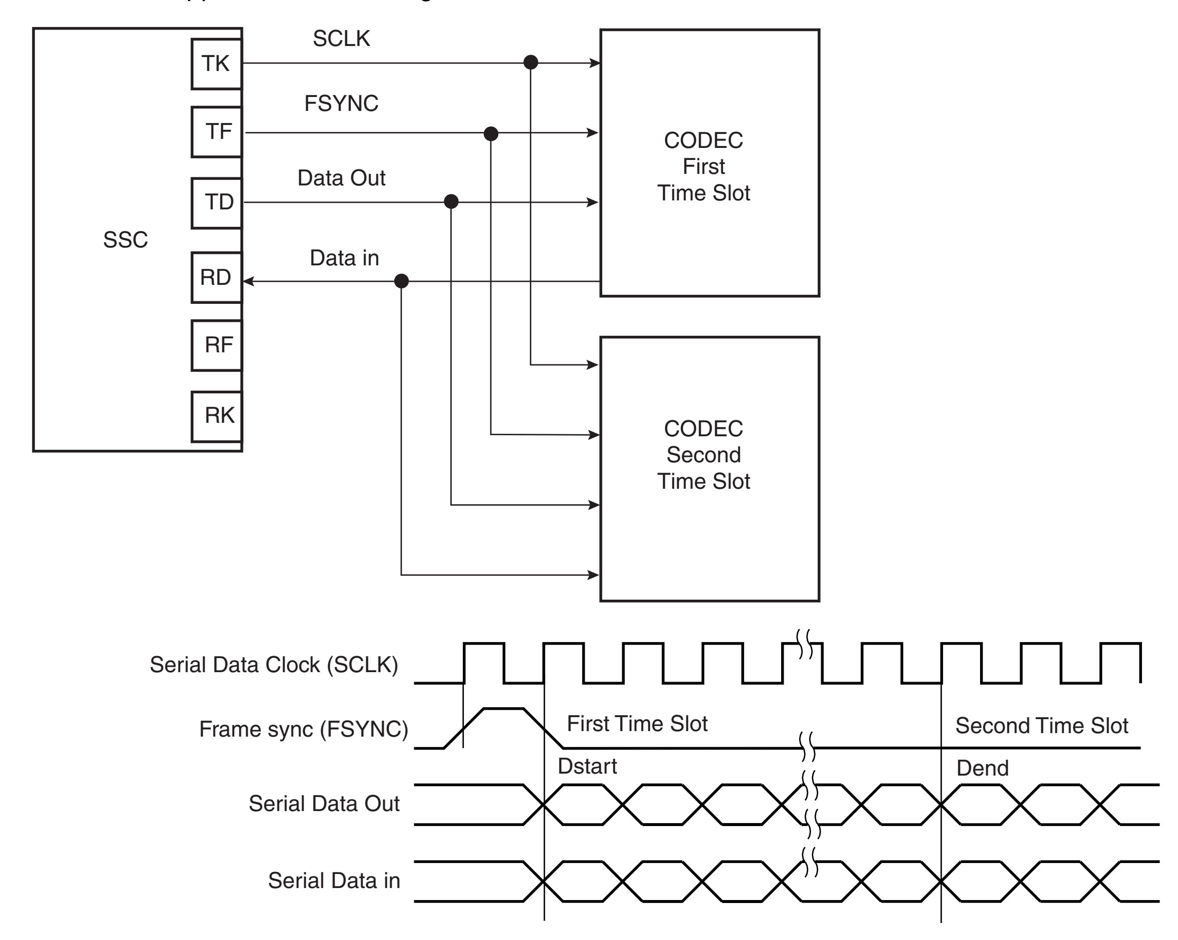


Figure 32-19. Time Slot Application Block Diagram



32.8 Synchronous Serial Controller (SSC) User Interface

Table 32-4. Register Mapping

Offset	Register	Register Name	Access	Reset
0x0	Control Register	SSC_CR	Write	–
0x4	Clock Mode Register	SSC_CMR	Read/Write	0x0
0x8	Reserved	–	–	–
0xC	Reserved	–	–	–
0x10	Receive Clock Mode Register	SSC_RCMR	Read/Write	0x0
0x14	Receive Frame Mode Register	SSC_RFMR	Read/Write	0x0
0x18	Transmit Clock Mode Register	SSC_TCMR	Read/Write	0x0
0x1C	Transmit Frame Mode Register	SSC_TFMR	Read/Write	0x0
0x20	Receive Holding Register	SSC_RHR	Read	0x0
0x24	Transmit Holding Register	SSC_THR	Write	–
0x28	Reserved	–	–	–
0x2C	Reserved	–	–	–
0x30	Receive Sync. Holding Register	SSC_RSHR	Read	0x0
0x34	Transmit Sync. Holding Register	SSC_TSHR	Read/Write	0x0
0x38	Receive Compare 0 Register	SSC_RC0R	Read/Write	0x0
0x3C	Receive Compare 1 Register	SSC_RC1R	Read/Write	0x0
0x40	Status Register	SSC_SR	Read	0x000000CC
0x44	Interrupt Enable Register	SSC_IER	Write	–
0x48	Interrupt Disable Register	SSC_IDR	Write	–
0x4C	Interrupt Mask Register	SSC_IMR	Read	0x0
0x50-0xFC	Reserved	–	–	–
0x100- 0x124	Reserved for Peripheral Data Controller (PDC)	–	–	–



32.8.1 SSC Control Register

Name: SSC_CR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SWRST	–	–	–	–	–	TXDIS	TXEN
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	RXDIS	RXEN

- **RXEN: Receive Enable**

0: No effect.

1: Enables Receive if RXDIS is not set.

- **RXDIS: Receive Disable**

0: No effect.

1: Disables Receive. If a character is currently being received, disables at end of current character reception.

- **TXEN: Transmit Enable**

0: No effect.

1: Enables Transmit if TXDIS is not set.

- **TXDIS: Transmit Disable**

0: No effect.

1: Disables Transmit. If a character is currently being transmitted, disables at end of current character transmission.

- **SWRST: Software Reset**

0: No effect.

1: Performs a software reset. Has priority on any other bit in SSC_CR.

32.8.2 SSC Clock Mode Register

Name: SSC_CMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	DIV			

7	6	5	4	3	2	1	0
DIV							

- **DIV: Clock Divider**

0: The Clock Divider is not active.

Any Other Value: The Divided Clock equals the Master Clock divided by 2 times DIV. The maximum bit rate is MCK/2. The minimum bit rate is MCK/2 x 4095 = MCK/8190.

32.8.3 SSC Receive Clock Mode Register

Name: SSC_RCMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
PERIOD							

23	22	21	20	19	18	17	16
STDDLY							

15	14	13	12	11	10	9	8
–	–	–	STOP	START			

7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

- **CKS: Receive Clock Selection**

CKS	Selected Receive Clock
0x0	Divided Clock
0x1	TK Clock signal
0x2	RK pin
0x3	Reserved

- **CKO: Receive Clock Output Mode Selection**

CKO	Receive Clock Output Mode	RK pin
0x0	None	Input-only
0x1	Continuous Receive Clock	Output
0x2	Receive Clock only during data transfers	Output
0x3-0x7	Reserved	

- **CKI: Receive Clock Inversion**

0: The data inputs (Data and Frame Sync signals) are sampled on Receive Clock falling edge. The Frame Sync signal output is shifted out on Receive Clock rising edge.

1: The data inputs (Data and Frame Sync signals) are sampled on Receive Clock rising edge. The Frame Sync signal output is shifted out on Receive Clock falling edge.

CKI affects only the Receive Clock and not the output clock signal.

- **CKG: Receive Clock Gating Selection**

CKG	Receive Clock Gating
0x0	None, continuous clock
0x1	Receive Clock enabled only if RF Low
0x2	Receive Clock enabled only if RF High
0x3	Reserved

- **START: Receive Start Selection**

START	Receive Start
0x0	Continuous, as soon as the receiver is enabled, and immediately after the end of transfer of the previous data.
0x1	Transmit start
0x2	Detection of a low level on RF signal
0x3	Detection of a high level on RF signal
0x4	Detection of a falling edge on RF signal
0x5	Detection of a rising edge on RF signal
0x6	Detection of any level change on RF signal
0x7	Detection of any edge on RF signal
0x8	Compare 0
0x9-0xF	Reserved

- **STOP: Receive Stop Selection**

0: After completion of a data transfer when starting with a Compare 0, the receiver stops the data transfer and waits for a new compare 0.

1: After starting a receive with a Compare 0, the receiver operates in a continuous mode until a Compare 1 is detected.

- **STTDLY: Receive Start Delay**

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of reception. When the Receiver is programmed to start synchronously with the Transmitter, the delay is also applied.

Note: It is very important that STTDLY be set carefully. If STTDLY must be set, it should be done in relation to TAG (Receive Sync Data) reception.

- **PERIOD: Receive Period Divider Selection**

This field selects the divider to apply to the selected Receive Clock in order to generate a new Frame Sync Signal. If 0, no PERIOD signal is generated. If not 0, a PERIOD signal is generated each 2 x (PERIOD+1) Receive Clock.



32.8.4 SSC Receive Frame Mode Register

Name: SSC_RFMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	FSEDGE

23	22	21	20	19	18	17	16
–	FSOS			FSLEN			

15	14	13	12	11	10	9	8
–	–	–	–	DATNB			

7	6	5	4	3	2	1	0
MSBF	–	LOOP	DATLEN				

- **DATLEN: Data Length**

0: Forbidden value (1-bit data length not supported).

Any other value: The bit stream contains DATLEN + 1 data bits. Moreover, it defines the transfer size performed by the PDC2 assigned to the Receiver. If DATLEN is lower or equal to 7, data transfers are in bytes. If DATLEN is between 8 and 15 (included), half-words are transferred, and for any other value, 32-bit words are transferred.

- **LOOP: Loop Mode**

0: Normal operating mode.

1: RD is driven by TD, RF is driven by TF and TK drives RK.

- **MSBF: Most Significant Bit First**

0: The lowest significant bit of the data register is sampled first in the bit stream.

1: The most significant bit of the data register is sampled first in the bit stream.

- **DATNB: Data Number per Frame**

This field defines the number of data words to be received after each transfer start, which is equal to (DATNB + 1).

- **FSLEN: Receive Frame Sync Length**

This field defines the length of the Receive Frame Sync Signal and the number of bits sampled and stored in the Receive Sync Data Register. When this mode is selected by the START field in the Receive Clock Mode Register, it also determines the length of the sampled data to be compared to the Compare 0 or Compare 1 register.

Pulse length is equal to (FSLEN + 1) Receive Clock periods. Thus, if FSLEN is 0, the Receive Frame Sync signal is generated during one Receive Clock period.

- **FSOS: Receive Frame Sync Output Selection**

FSOS	Selected Receive Frame Sync Signal	RF Pin
0x0	None	Input-only
0x1	Negative Pulse	Output
0x2	Positive Pulse	Output
0x3	Driven Low during data transfer	Output
0x4	Driven High during data transfer	Output
0x5	Toggling at each start of data transfer	Output
0x6-0x7	Reserved	Undefined

- **FSEDGE: Frame Sync Edge Detection**

Determines which edge on Frame Sync will generate the interrupt RXSYN in the SSC Status Register.

FSEDGE	Frame Sync Edge Detection
0x0	Positive Edge Detection
0x1	Negative Edge Detection



32.8.5 SSC Transmit Clock Mode Register

Name: SSC_TCMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
PERIOD							

23	22	21	20	19	18	17	16
STTDLY							

15	14	13	12	11	10	9	8
–	–	–	–	START			

7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

- **CKS: Transmit Clock Selection**

CKS	Selected Transmit Clock
0x0	Divided Clock
0x1	RK Clock signal
0x2	TK Pin
0x3	Reserved

- **CKO: Transmit Clock Output Mode Selection**

CKO	Transmit Clock Output Mode	TK pin
0x0	None	Input-only
0x1	Continuous Transmit Clock	Output
0x2	Transmit Clock only during data transfers	Output
0x3-0x7	Reserved	

- **CKI: Transmit Clock Inversion**

0: The data outputs (Data and Frame Sync signals) are shifted out on Transmit Clock falling edge. The Frame sync signal input is sampled on Transmit clock rising edge.

1: The data outputs (Data and Frame Sync signals) are shifted out on Transmit Clock rising edge. The Frame sync signal input is sampled on Transmit clock falling edge.

CKI affects only the Transmit Clock and not the output clock signal.

- **CKG: Transmit Clock Gating Selection**

CKG	Transmit Clock Gating
0x0	None, continuous clock
0x1	Transmit Clock enabled only if TF Low
0x2	Transmit Clock enabled only if TF High
0x3	Reserved

- **START: Transmit Start Selection**

START	Transmit Start
0x0	Continuous, as soon as a word is written in the SSC_THR Register (if Transmit is enabled), and immediately after the end of transfer of the previous data.
0x1	Receive start
0x2	Detection of a low level on TF signal
0x3	Detection of a high level on TF signal
0x4	Detection of a falling edge on TF signal
0x5	Detection of a rising edge on TF signal
0x6	Detection of any level change on TF signal
0x7	Detection of any edge on TF signal
0x8 - 0xF	Reserved

- **STTDLY: Transmit Start Delay**

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of transmission of data. When the Transmitter is programmed to start synchronously with the Receiver, the delay is also applied.

Note: STTDLY must be set carefully. If STTDLY is too short in respect to TAG (Transmit Sync Data) emission, data is emitted instead of the end of TAG.

- **PERIOD: Transmit Period Divider Selection**

This field selects the divider to apply to the selected Transmit Clock to generate a new Frame Sync Signal. If 0, no period signal is generated. If not 0, a period signal is generated at each 2 x (PERIOD+1) Transmit Clock.

32.8.6 SSC Transmit Frame Mode Register

Name: SSC_TFMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	FSEDGE

23	22	21	20	19	18	17	16
FSDEN	FSOS			FSLEN			

15	14	13	12	11	10	9	8
–	–	–	–	DATNB			

7	6	5	4	3	2	1	0
MSBF	–	DATDEF	DATLEN				

- **DATLEN: Data Length**

0: Forbidden value (1-bit data length not supported).

Any other value: The bit stream contains DATLEN + 1 data bits. Moreover, it defines the transfer size performed by the PDC2 assigned to the Transmit. If DATLEN is lower or equal to 7, data transfers are bytes, if DATLEN is between 8 and 15 (included), half-words are transferred, and for any other value, 32-bit words are transferred.

- **DATDEF: Data Default Value**

This bit defines the level driven on the TD pin while out of transmission. Note that if the pin is defined as multi-drive by the PIO Controller, the pin is enabled only if the SCC TD output is 1.

- **MSBF: Most Significant Bit First**

0: The lowest significant bit of the data register is shifted out first in the bit stream.

1: The most significant bit of the data register is shifted out first in the bit stream.

- **DATNB: Data Number per frame**

This field defines the number of data words to be transferred after each transfer start, which is equal to (DATNB +1).

- **FSLEN: Transmit Frame Sync Length**

This field defines the length of the Transmit Frame Sync signal and the number of bits shifted out from the Transmit Sync Data Register if FSDEN is 1.

Pulse length is equal to (FSLEN + 1) Transmit Clock periods, i.e., the pulse length can range from 1 to 16 Transmit Clock periods. If FSLEN is 0, the Transmit Frame Sync signal is generated during one Transmit Clock period.

- **FSOS: Transmit Frame Sync Output Selection**

FSOS	Selected Transmit Frame Sync Signal	TF Pin
0x0	None	Input-only
0x1	Negative Pulse	Output
0x2	Positive Pulse	Output
0x3	Driven Low during data transfer	Output
0x4	Driven High during data transfer	Output
0x5	Toggling at each start of data transfer	Output
0x6-0x7	Reserved	Undefined

- **FSDEN: Frame Sync Data Enable**

0: The TD line is driven with the default value during the Transmit Frame Sync signal.

1: SSC_TSHR value is shifted out during the transmission of the Transmit Frame Sync signal.

- **FSEDGE: Frame Sync Edge Detection**

Determines which edge on frame sync will generate the interrupt TXSYN (Status Register).

FSEDGE	Frame Sync Edge Detection
0x0	Positive Edge Detection
0x1	Negative Edge Detection

32.8.7 SSC Receive Holding Register

Name: SSC_RHR

Access Type: Read-only

31	30	29	28	27	26	25	24
RDAT							

23	22	21	20	19	18	17	16
RDAT							

15	14	13	12	11	10	9	8
RDAT							

7	6	5	4	3	2	1	0
RDAT							

- **RDAT: Receive Data**

Right aligned regardless of the number of data bits defined by DATLEN in SSC_RFMR.

32.8.8 SSC Transmit Holding Register

Name: SSC_THR

Access Type: Write-only

31	30	29	28	27	26	25	24
TDAT							

23	22	21	20	19	18	17	16
TDAT							

15	14	13	12	11	10	9	8
TDAT							

7	6	5	4	3	2	1	0
TDAT							

- **TDAT: Transmit Data**

Right aligned regardless of the number of data bits defined by DATLEN in SSC_TFMR.

32.8.9 SSC Receive Synchronization Holding Register

Name: SSC_RSHR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
RSDAT							

7	6	5	4	3	2	1	0
RSDAT							

- **RSDAT: Receive Synchronization Data**

32.8.10 SSC Transmit Synchronization Holding Register

Name: SSC_TSHR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
TSDAT							

7	6	5	4	3	2	1	0
TSDAT							

- **TSDAT: Transmit Synchronization Data**

32.8.11 SSC Receive Compare 0 Register

Name: SSC_RC0R

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
CP0							

7	6	5	4	3	2	1	0
CP0							

- **CP0: Receive Compare Data 0**

32.8.12 SSC Receive Compare 1 Register

Name: SSC_RC1R

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
CP1							

7	6	5	4	3	2	1	0
CP1							

- **CP1: Receive Compare Data 1**

32.8.13 SSC Status Register

Name: SSC_SR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	RXEN	TXEN

15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0

7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready**

0: Data has been loaded in SSC_THR and is waiting to be loaded in the Transmit Shift Register (TSR).

1: SSC_THR is empty.

- **TXEMPTY: Transmit Empty**

0: Data remains in SSC_THR or is currently transmitted from TSR.

1: Last data written in SSC_THR has been loaded in TSR and last data loaded in TSR has been transmitted.

- **ENDTX: End of Transmission**

0: The register SSC_TCR has not reached 0 since the last write in SSC_TCR or SSC_TNCR.

1: The register SSC_TCR has reached 0 since the last write in SSC_TCR or SSC_TNCR.

- **TXBUFE: Transmit Buffer Empty**

0: SSC_TCR or SSC_TNCR have a value other than 0.

1: Both SSC_TCR and SSC_TNCR have a value of 0.

- **RXRDY: Receive Ready**

0: SSC_RHR is empty.

1: Data has been received and loaded in SSC_RHR.

- **OVRUN: Receive Overrun**

0: No data has been loaded in SSC_RHR while previous data has not been read since the last read of the Status Register.

1: Data has been loaded in SSC_RHR while previous data has not yet been read since the last read of the Status Register.

- **ENDRX: End of Reception**

0: Data is written on the Receive Counter Register or Receive Next Counter Register.

1: End of PDC transfer when Receive Counter Register has arrived at zero.

- **RXBUFF: Receive Buffer Full**

0: SSC_RCR or SSC_RNCR have a value other than 0.

1: Both SSC_RCR and SSC_RNCR have a value of 0.

- **CP0: Compare 0**

0: A compare 0 has not occurred since the last read of the Status Register.

1: A compare 0 has occurred since the last read of the Status Register.

- **CP1: Compare 1**

0: A compare 1 has not occurred since the last read of the Status Register.

1: A compare 1 has occurred since the last read of the Status Register.



- **TXSYN: Transmit Sync**

0: A Tx Sync has not occurred since the last read of the Status Register.

1: A Tx Sync has occurred since the last read of the Status Register.

- **RXSYN: Receive Sync**

0: An Rx Sync has not occurred since the last read of the Status Register.

1: An Rx Sync has occurred since the last read of the Status Register.

- **TXEN: Transmit Enable**

0: Transmit is disabled.

1: Transmit is enabled.

- **RXEN: Receive Enable**

0: Receive is disabled.

1: Receive is enabled.

32.8.14 SSC Interrupt Enable Register

Name: SSC_IER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0

7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Enable**

0: No effect.

1: Enables the Transmit Ready Interrupt.

- **TXEMPTY: Transmit Empty Interrupt Enable**

0: No effect.

1: Enables the Transmit Empty Interrupt.

- **ENDTX: End of Transmission Interrupt Enable**

0: No effect.

1: Enables the End of Transmission Interrupt.

- **TXBUFE: Transmit Buffer Empty Interrupt Enable**

0: No effect.

1: Enables the Transmit Buffer Empty Interrupt

- **RXRDY: Receive Ready Interrupt Enable**

0: No effect.

1: Enables the Receive Ready Interrupt.

- **OVRUN: Receive Overrun Interrupt Enable**

0: No effect.

1: Enables the Receive Overrun Interrupt.

- **ENDRX: End of Reception Interrupt Enable**

0: No effect.

1: Enables the End of Reception Interrupt.

- **RXBUFF: Receive Buffer Full Interrupt Enable**

0: No effect.

1: Enables the Receive Buffer Full Interrupt.

- **CP0: Compare 0 Interrupt Enable**

0: No effect.

1: Enables the Compare 0 Interrupt.

- **CP1: Compare 1 Interrupt Enable**

0: No effect.

1: Enables the Compare 1 Interrupt.

- **TXSYN: Tx Sync Interrupt Enable**

0: No effect.

1: Enables the Tx Sync Interrupt.

- **RXSYN: Rx Sync Interrupt Enable**

0: No effect.

1: Enables the Rx Sync Interrupt.

32.8.15 SSC Interrupt Disable Register

Name: SSC_IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0

7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Disable**

0: No effect.

1: Disables the Transmit Ready Interrupt.

- **TXEMPTY: Transmit Empty Interrupt Disable**

0: No effect.

1: Disables the Transmit Empty Interrupt.

- **ENDTX: End of Transmission Interrupt Disable**

0: No effect.

1: Disables the End of Transmission Interrupt.

- **TXBUFE: Transmit Buffer Empty Interrupt Disable**

0: No effect.

1: Disables the Transmit Buffer Empty Interrupt.

- **RXRDY: Receive Ready Interrupt Disable**

0: No effect.

1: Disables the Receive Ready Interrupt.

- **OVRUN: Receive Overrun Interrupt Disable**

0: No effect.

1: Disables the Receive Overrun Interrupt.

- **ENDRX: End of Reception Interrupt Disable**

0: No effect.

1: Disables the End of Reception Interrupt.

- **RXBUFF: Receive Buffer Full Interrupt Disable**

0: No effect.

1: Disables the Receive Buffer Full Interrupt.

- **CP0: Compare 0 Interrupt Disable**

0: No effect.

1: Disables the Compare 0 Interrupt.

- **CP1: Compare 1 Interrupt Disable**

0: No effect.

1: Disables the Compare 1 Interrupt.

- **TXSYN: Tx Sync Interrupt Enable**

0: No effect.

1: Disables the Tx Sync Interrupt.

- **RXSYN: Rx Sync Interrupt Enable**

0: No effect.

1: Disables the Rx Sync Interrupt.

32.8.16 SSC Interrupt Mask Register

Name: SSC_IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0

7	6	5	4	3	2	1	0
RXBUF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Mask**

0: The Transmit Ready Interrupt is disabled.

1: The Transmit Ready Interrupt is enabled.

- **TXEMPTY: Transmit Empty Interrupt Mask**

0: The Transmit Empty Interrupt is disabled.

1: The Transmit Empty Interrupt is enabled.

- **ENDTX: End of Transmission Interrupt Mask**

0: The End of Transmission Interrupt is disabled.

1: The End of Transmission Interrupt is enabled.

- **TXBUFE: Transmit Buffer Empty Interrupt Mask**

0: The Transmit Buffer Empty Interrupt is disabled.

1: The Transmit Buffer Empty Interrupt is enabled.

- **RXRDY: Receive Ready Interrupt Mask**

0: The Receive Ready Interrupt is disabled.

1: The Receive Ready Interrupt is enabled.

- **OVRUN: Receive Overrun Interrupt Mask**

0: The Receive Overrun Interrupt is disabled.

1: The Receive Overrun Interrupt is enabled.

- **ENDRX: End of Reception Interrupt Mask**

0: The End of Reception Interrupt is disabled.

1: The End of Reception Interrupt is enabled.

- **RXBUFF: Receive Buffer Full Interrupt Mask**

0: The Receive Buffer Full Interrupt is disabled.

1: The Receive Buffer Full Interrupt is enabled.

- **CP0: Compare 0 Interrupt Mask**

0: The Compare 0 Interrupt is disabled.

1: The Compare 0 Interrupt is enabled.

- **CP1: Compare 1 Interrupt Mask**

0: The Compare 1 Interrupt is disabled.

1: The Compare 1 Interrupt is enabled.



- **TXSYN: Tx Sync Interrupt Mask**

0: The Tx Sync Interrupt is disabled.

1: The Tx Sync Interrupt is enabled.

- **RXSYN: Rx Sync Interrupt Mask**

0: The Rx Sync Interrupt is disabled.

1: The Rx Sync Interrupt is enabled.

33. Timer/Counter (TC)

33.1 Overview

The Timer/Counter (TC) includes three identical 16-bit Timer/Counter channels.

Each channel can be independently programmed to perform a wide range of functions including frequency measurement, event counting, interval measurement, pulse generation, delay timing and pulse width modulation.

Each channel has three external clock inputs, five internal clock inputs and two multi-purpose input/output signals which can be configured by the user. Each channel drives an internal interrupt signal which can be programmed to generate processor interrupts.

The Timer/Counter block has two global registers which act upon all three TC channels.

The Block Control Register allows the three channels to be started simultaneously with the same instruction.

The Block Mode Register defines the external clock inputs for each channel, allowing them to be chained.

33.2 Block Diagram

Figure 33-1. Timer/Counter Block Diagram

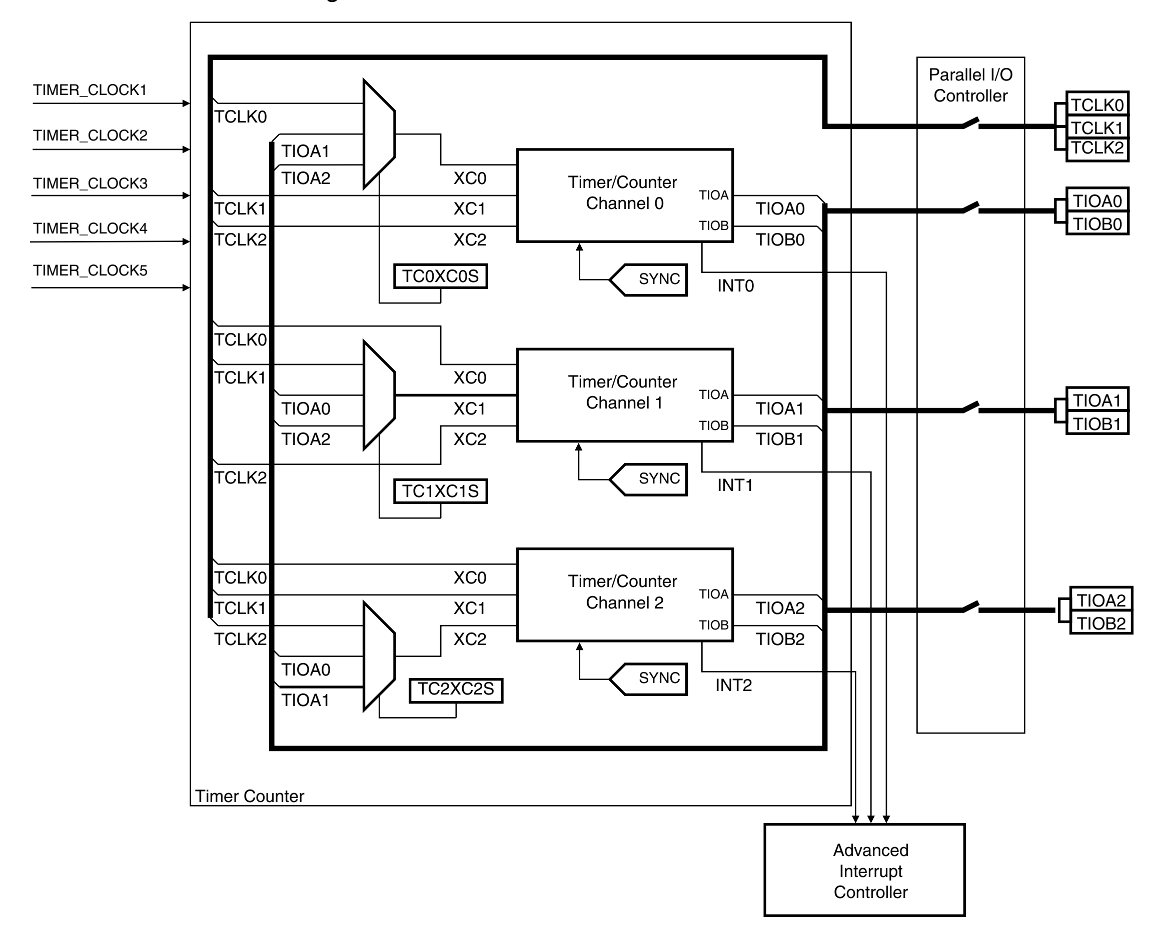


Table 33-1. Signal Name Description

Block/Channel	Signal Name	Description
Channel Signal	XC0, XC1, XC2	External Clock Inputs
	TIOA	Capture Mode: Timer/Counter Input Waveform Mode: Timer/Counter Output
	TIOB	Capture Mode: Timer/Counter Input Waveform Mode: Timer/Counter Input/output
	INT	Interrupt Signal Output
	SYNC	Synchronization Input Signal

33.3 Pin Name List

Table 33-2. TC Pin List

Pin Name	Description	Type
TCLK0-TCLK2	External Clock Input	Input
TIOA0-TIOA2	I/O Line A	I/O
TIOB0-TIOB2	I/O Line B	I/O

33.4 Product Dependencies

33.4.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the TC pins to their peripheral functions.

33.4.2 Power Management

The TC is clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the Timer/Counter clock.

33.4.3 Interrupt

The TC has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling the TC interrupt requires programming the AIC before configuring the TC.

33.5 Functional Description

33.5.1 TC Description

The three channels of the Timer/Counter are independent and identical in operation. The registers for channel programming are listed in Table 33-4 on page 476.

33.5.1.1 16-bit Counter

Each channel is organized around a 16-bit counter. The value of the counter is incremented at each positive edge of the selected clock. When the counter has reached the value 0xFFFF and passes to 0x0000, an overflow occurs and the COVFS bit in TC_SR (Status Register) is set.

The current value of the counter is accessible in real time by reading the Counter Value Register, TC_CV. The counter can be reset by a trigger. In this case, the counter value passes to 0x0000 on the next valid edge of the selected clock.

33.5.1.2 Clock Selection

At block level, input clock signals of each channel can either be connected to the external inputs TCLK0, TCLK1 or TCLK2, or be connected to the configurable I/O signals TIOA0, TIOA1 or TIOA2 for chaining by programming the TC_BMR (Block Mode). See Figure 33-2.

Each channel can independently select an internal or external clock source for its counter:

- Internal clock signals: TIMER_CLOCK1, TIMER_CLOCK2, TIMER_CLOCK3, TIMER_CLOCK4, TIMER_CLOCK5
- External clock signals: XC0, XC1 or XC2

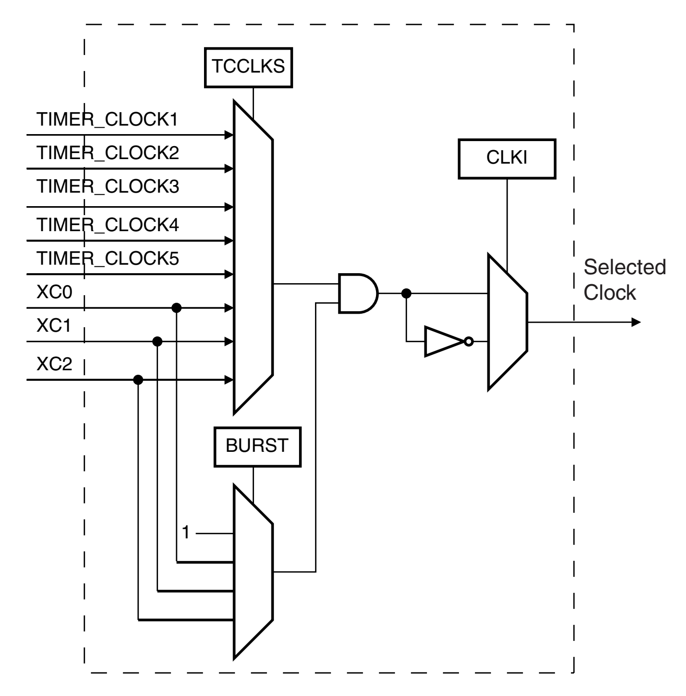
This selection is made by the TCCLKS bits in the TC Channel Mode Register.

The selected clock can be inverted with the CLKI bit in TC_CMR. This allows counting on the opposite edges of the clock.

The burst function allows the clock to be validated when an external signal is high. The BURST parameter in the Mode Register defines this signal (none, XC0, XC1, XC2).

Note: In all cases, if an external clock is used, the duration of each of its levels must be longer than the master clock period. The external clock frequency must be at least 2.5 times lower than the master clock

Figure 33-2. Clock Selection

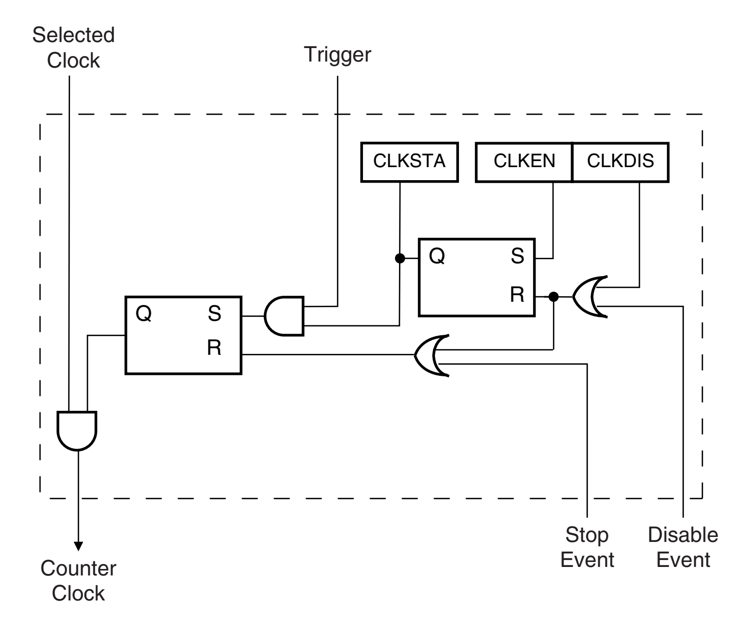


33.5.1.3 Clock Control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 33-3.

- The clock can be enabled or disabled by the user with the CLKEN and the CLKDIS commands in the Control Register. In Capture Mode it can be disabled by an RB load event if LDBDIS is set to 1 in TC_CMR. In Waveform Mode, it can be disabled by an RC Compare event if CPCDIS is set to 1 in TC_CMR. When disabled, the start or the stop actions have no effect: only a CLKEN command in the Control Register can re-enable the clock. When the clock is enabled, the CLKSTA bit is set in the Status Register.
- The clock can also be started or stopped: a trigger (software, synchro, external or compare) always starts the clock. The clock can be stopped by an RB load event in Capture Mode (LDBSTOP = 1 in TC_CMR) or a RC compare event in Waveform Mode (CPCSTOP = 1 in TC_CMR). The start and the stop commands have effect only if the clock is enabled.

Figure 33-3. Clock Control



33.5.1.4 TC Operating Modes

Each channel can independently operate in two different modes:

- Capture Mode provides measurement on signals.
- Waveform Mode provides wave generation.

The TC Operating Mode is programmed with the WAVE bit in the TC Channel Mode Register.

In Capture Mode, TIOA and TIOB are configured as inputs.

In Waveform Mode, TIOA is always configured to be an output and TIOB is an output if it is not selected to be the external trigger.

33.5.1.5 Trigger

A trigger resets the counter and starts the counter clock. Three types of triggers are common to both modes, and a fourth external trigger is available to each mode.

The following triggers are common to both modes:

- Software Trigger: Each channel has a software trigger, available by setting SWTRG in TC_CCR.
- SYNC: Each channel has a synchronization signal SYNC. When asserted, this signal has the same effect as a software trigger. The SYNC signals of all channels are asserted simultaneously by writing TC_BCR (Block Control) with SYNC set.
- Compare RC Trigger: RC is implemented in each channel and can provide a trigger when the counter value matches the RC value if CPCTRG is set in TC_CMR.

The channel can also be configured to have an external trigger. In Capture Mode, the external trigger signal can be selected between TIOA and TIOB. In Waveform Mode, an external event can be programmed on one of the following signals: TIOB, XC0, XC1 or XC2. This external event can then be programmed to perform a trigger by setting ENETRG in TC_CMR.

If an external trigger is used, the duration of the pulses must be longer than the master clock period in order to be detected.

Regardless of the trigger used, it will be taken into account at the following active edge of the selected clock. This means that the counter value can be read differently from zero just after a trigger, especially when a low frequency signal is selected as the clock.

33.5.2 Capture Operating Mode

This mode is entered by clearing the WAVE parameter in TC_CMR (Channel Mode Register).

Capture Mode allows the TC channel to perform measurements such as pulse timing, frequency, period, duty cycle and phase on TIOA and TIOB signals which are considered as inputs.

Figure 33-4 shows the configuration of the TC channel when programmed in Capture Mode.

33.5.2.1 Capture Registers A and B

Registers A and B (RA and RB) are used as capture registers. This means that they can be loaded with the counter value when a programmable event occurs on the signal TIOA.

The LDRA parameter in TC_CMR defines the TIOA edge for the loading of register A, and the LDRB parameter defines the TIOA edge for the loading of Register B.

RA is loaded only if it has not been loaded since the last trigger or if RB has been loaded since the last loading of RA.

RB is loaded only if RA has been loaded since the last trigger or the last loading of RB.

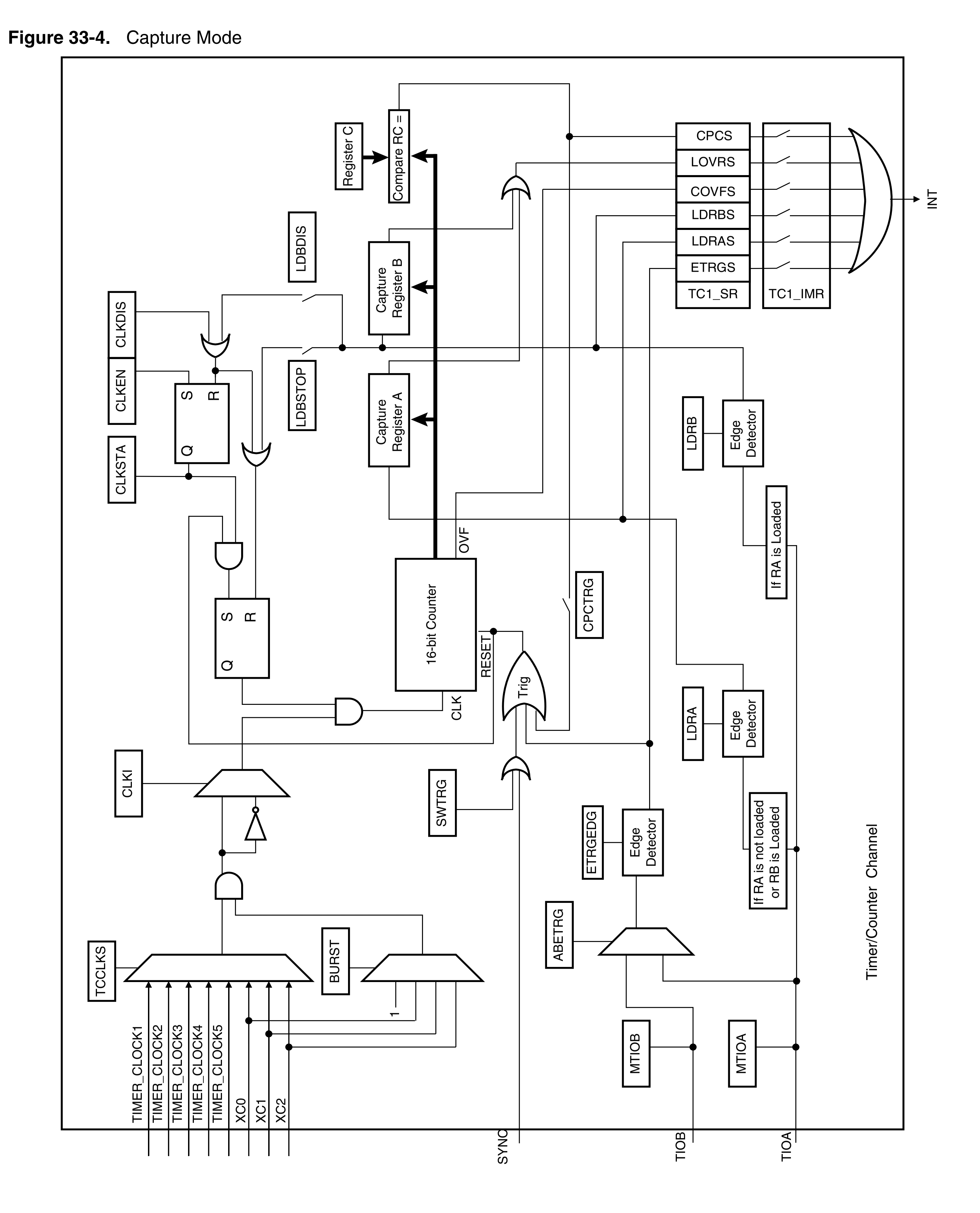
Loading RA or RB before the read of the last value loaded sets the Overrun Error Flag (LOVRS) in TC_SR (Status Register). In this case, the old value is overwritten.

33.5.2.2 Trigger Conditions

In addition to the SYNC signal, the software trigger and the RC compare trigger, an external trigger can be defined.

The ABETRG bit in TC_CMR selects TIOA or TIOB input signal as an external trigger. The ETRGEDG parameter defines the edge (rising, falling or both) detected to generate an external trigger. If ETRGEDG = 0 (none), the external trigger is disabled.

Figure 33-4. Capture Mode



33.5.3 Waveform Operating Mode

Waveform operating mode is entered by setting the WAVE parameter in TC_CMR (Channel Mode Register).

In Waveform Operating Mode the TC channel generates 1 or 2 PWM signals with the same frequency and independently programmable duty cycles, or generates different types of one-shot or repetitive pulses.

In this mode, TIOA is configured as an output and TIOB is defined as an output if it is not used as an external event (EEVT parameter in TC_CMR).

Figure 33-5 shows the configuration of the TC channel when programmed in Waveform Operating Mode.

33.5.3.1 Waveform Selection

Depending on the WAVSEL parameter in TC_CMR (Channel Mode Register), the behavior of TC_CV varies.

With any selection, RA, RB and RC can all be used as compare registers.

RA Compare is used to control the TIOA output, RB Compare is used to control the TIOB output (if correctly configured) and RC Compare is used to control TIOA and/or TIOB outputs.

Figure 33-5. Waveform Mode



WAVSEL = 00

When WAVSEL = 00, the value of TC_CV is incremented from 0 to 0xFFFF. Once 0xFFFF has been reached, the value of TC_CV is reset. Incrementation of TC_CV starts again and the cycle continues. See Figure 33-6.

An external event trigger or a software trigger can reset the value of TC_CV. It is important to note that the trigger may occur at any time. See Figure 33-7.

RC Compare cannot be programmed to generate a trigger in this configuration. At the same time, RC Compare can stop the counter clock (CPCSTOP = 1 in TC_CMR) and/or disable the counter clock (CPCDIS = 1 in TC_CMR).

Figure 33-6. WAVSEL= 00 without trigger

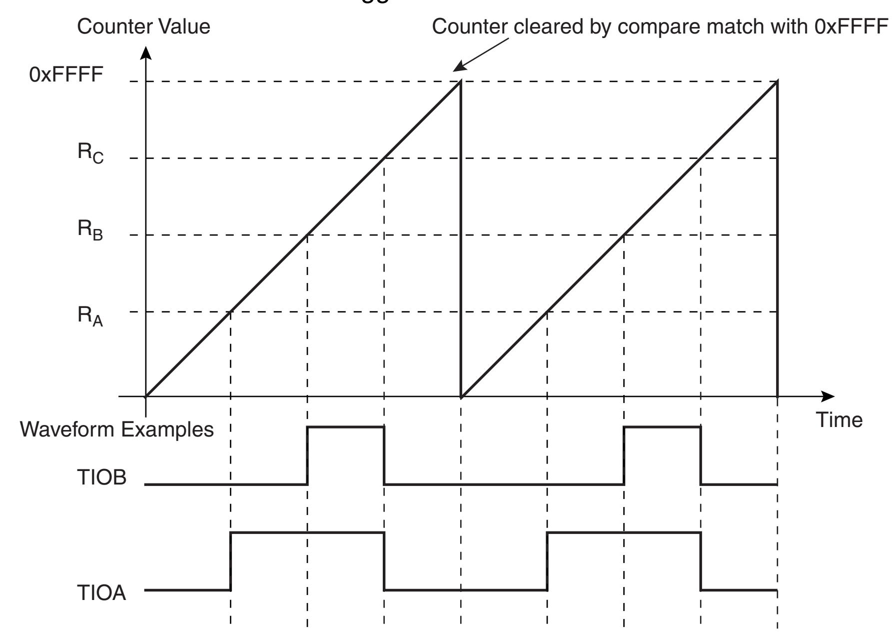


Figure 33-7. WAVSEL= 00 with trigger

Counter Value
Counter cleared by compare match with 0xFFFF
0xFFFF
R_C
Counter cleared by trigger
R_B
R_A
Time
Waveform Examples
TIOB
TIOA

WAVSEL = 10

When WAVSEL = 10, the value of TC_CV is incremented from 0 to the value of RC, then automatically reset on a RC Compare. Once the value of TC_CV has been reset, it is then incremented and so on. See Figure 33-8.

It is important to note that TC_CV can be reset at any time by an external event or a software trigger if both are programmed correctly. See Figure 33-9.

In addition, RC Compare can stop the counter clock (CPCSTOP = 1 in TC_CMR) and/or disable the counter clock (CPCDIS = 1 in TC_CMR).

Figure 33-8. WAVSEL = 10 Without Trigger

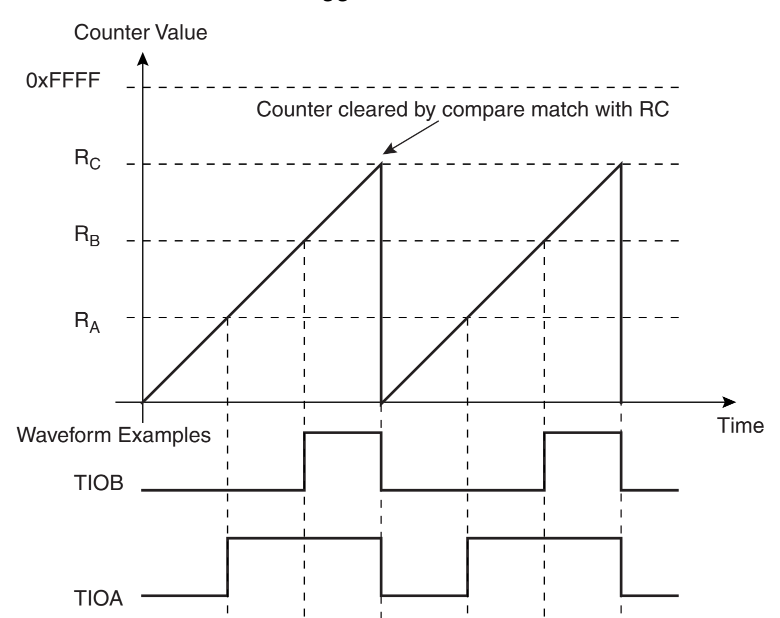
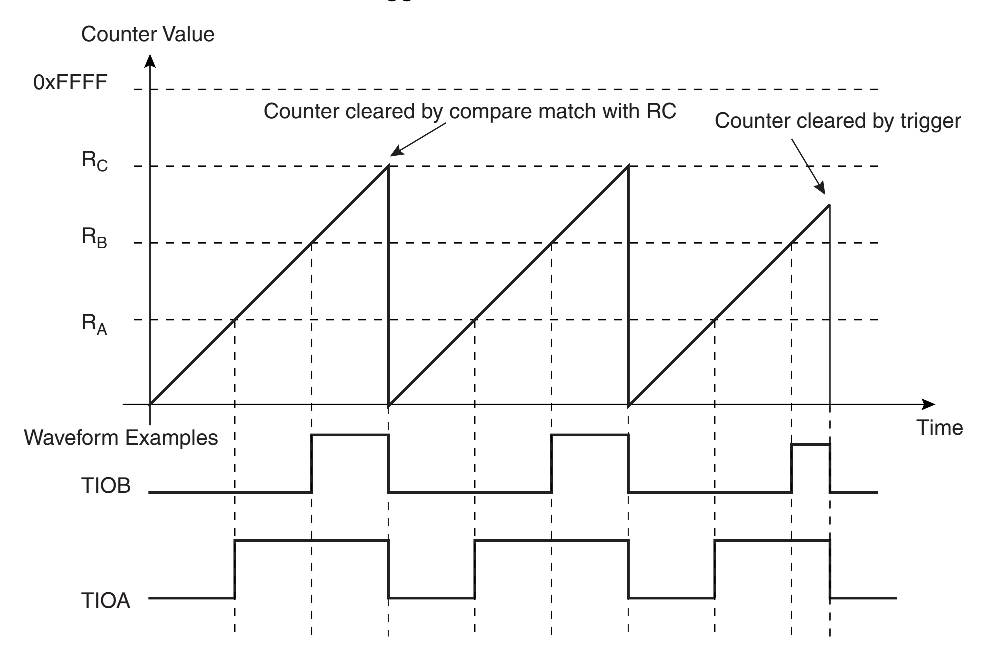




Figure 33-9. WAVSEL = 10 With Trigger



WAVSEL = 01

When WAVSEL = 01, the value of TC_CV is incremented from 0 to 0xFFFF. Once 0xFFFF is reached, the value of TC_CV is decremented to 0, then re-incremented to 0xFFFF and so on. See Figure 33-10.

A trigger such as an external event or a software trigger can modify TC_CV at any time. If a trigger occurs while TC_CV is incrementing, TC_CV then decrements. If a trigger is received while TC_CV is decrementing, TC_CV then increments. See Figure 33-11.

RC Compare cannot be programmed to generate a trigger in this configuration.

At the same time, RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).

Figure 33-10. WAVSEL = 01 Without Trigger

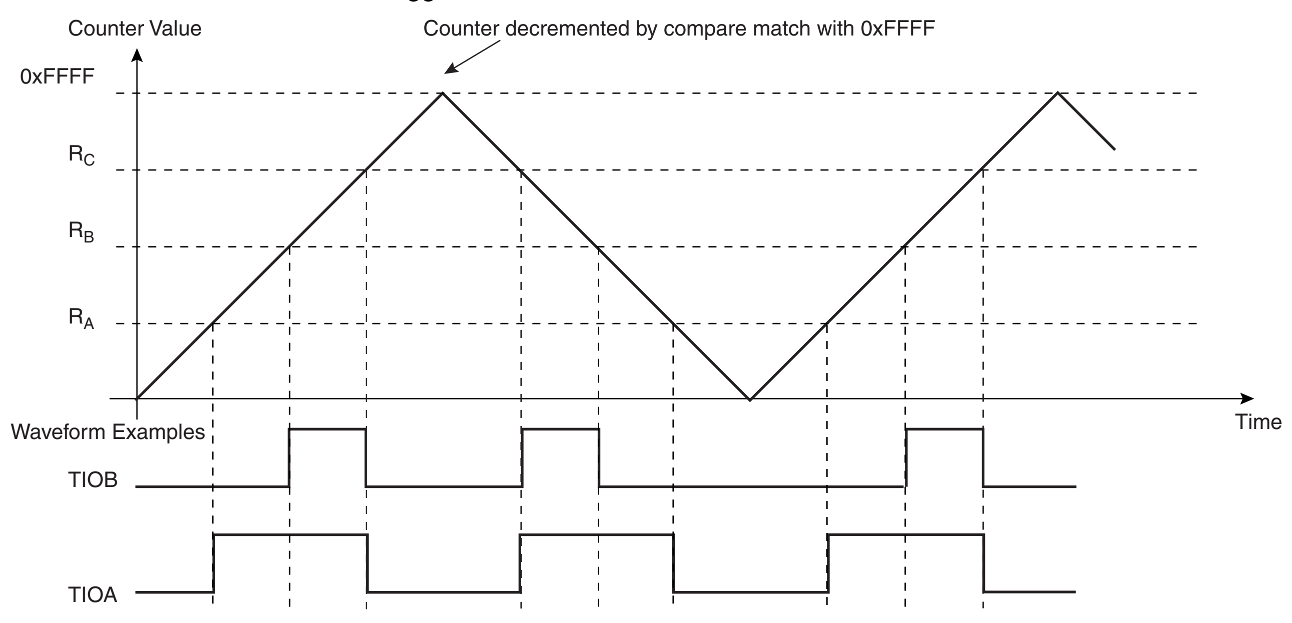
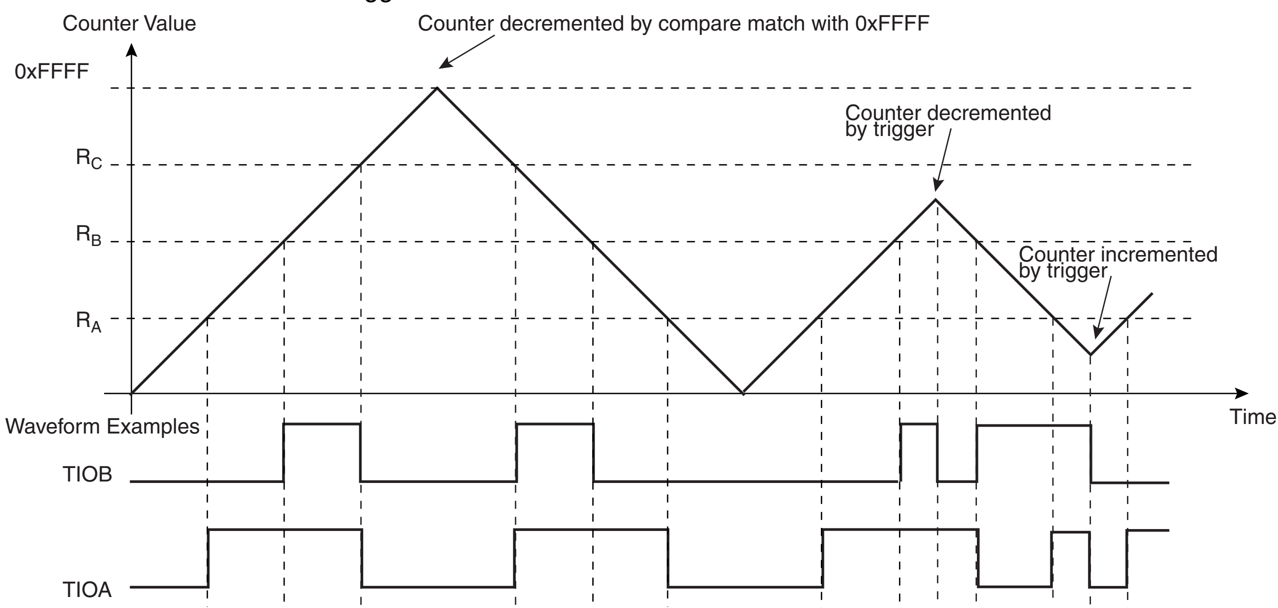


Figure 33-11. WAVSEL = 01 With Trigger



WAVSEL = 11

When WAVSEL = 11, the value of TC_CV is incremented from 0 to RC. Once RC is reached, the value of TC_CV is decremented to 0, then re-incremented to RC and so on. See Figure 33-12.

A trigger such as an external event or a software trigger can modify TC_CV at any time. If a trigger occurs while TC_CV is incrementing, TC_CV then decrements. If a trigger is received while TC_CV is decrementing, TC_CV then increments. See Figure 33-13.

RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).



Figure 33-12. WAVSEL = 11 Without Trigger

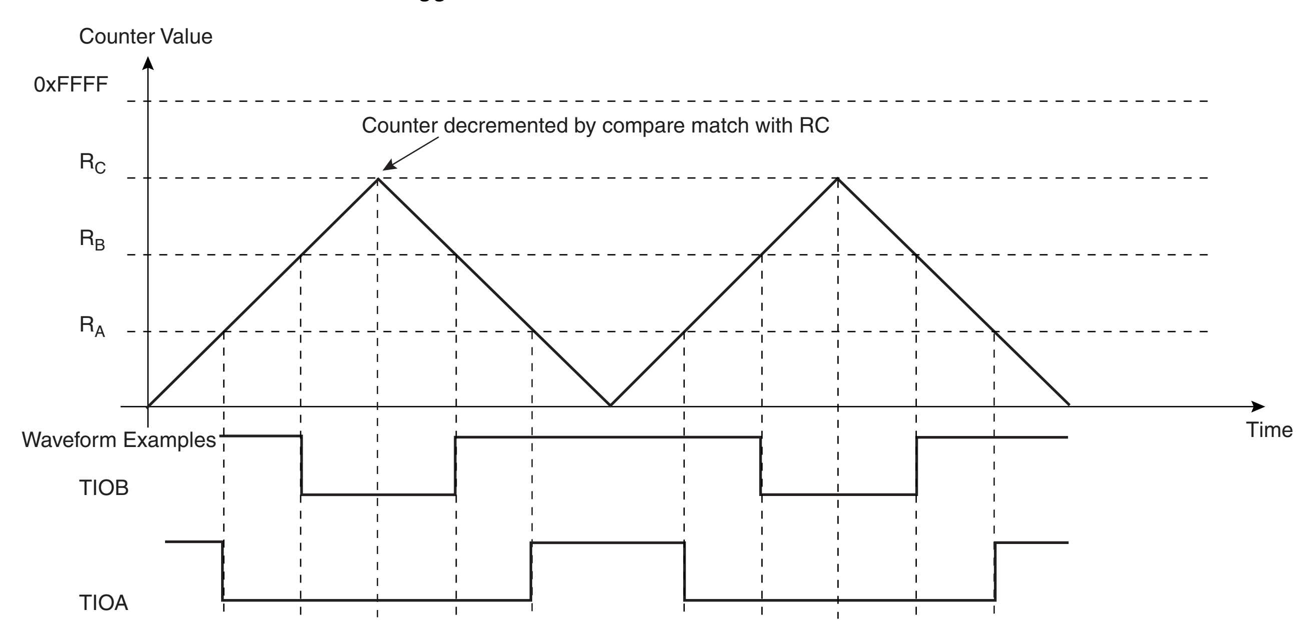
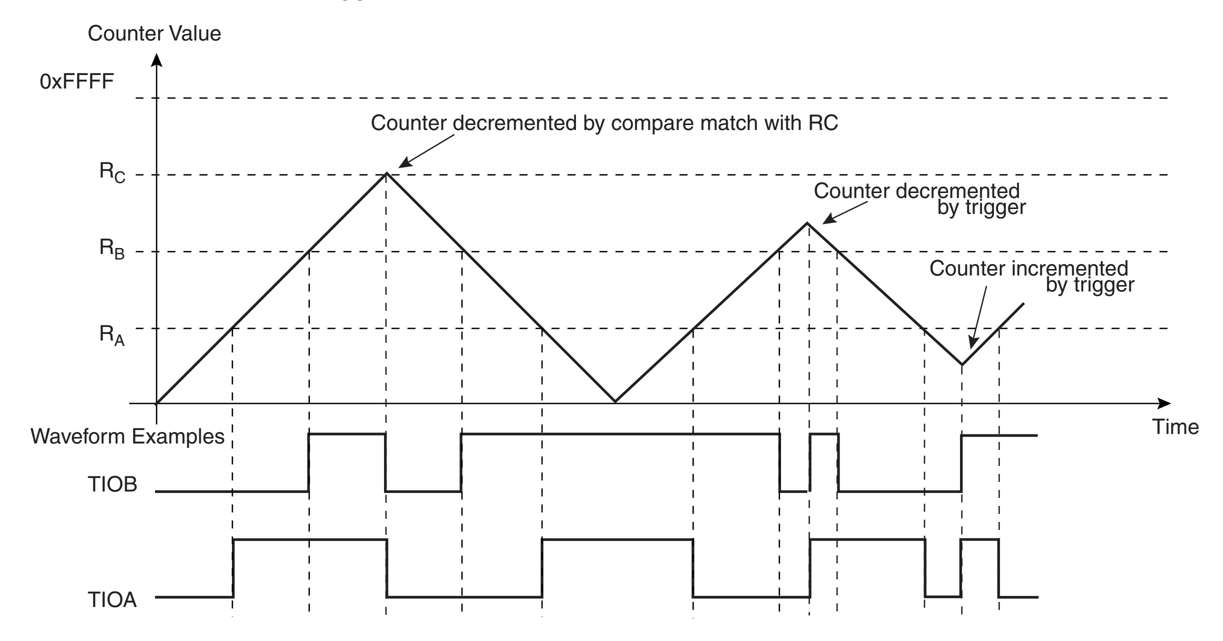


Figure 33-13. WAVSEL = 11 With Trigger



33.5.3.2 *External Event/Trigger Conditions*

An external event can be programmed to be detected on one of the clock sources (XC0, XC1, XC2) or TIOB. The external event selected can then be used as a trigger.

The parameter EEVT parameter in TC_CMR selects the external trigger. The EEVTEDG parameter defines the trigger edge for each of the possible external triggers (rising, falling or both). If EEVTEDG is cleared (none), no external event is defined.

If TIOB is defined as an external event signal (EEVT = 0), TIOB is no longer used as an output and the TC channel can only generate a waveform on TIOA.

When an external event is defined, it can be used as a trigger by setting bit ENETRG in TC_CMR.

As in Capture Mode, the SYNC signal and the software trigger are also available as triggers. RC Compare can also be used as a trigger depending on the parameter WAVSEL.

33.5.3.3 Output Controller

The output controller defines the output level changes on TIOA and TIOB following an event. TIOB control is used only if TIOB is defined as output (not as an external event).

The following events control TIOA and TIOB: software trigger, external event and RC compare. RA compare controls TIOA and RB compare controls TIOB. Each of these events can be programmed to set, clear or toggle the output as defined in the corresponding parameter in TC_CMR.



33.6 Timer/Counter (TC) User Interface

33.6.1 Global Register Mapping

Table 33-3. Timer/Counter (TC) Global Register Map

Offset	Channel/Register	Name	Access	Reset Value
0x00	TC Channel 0	See Table 33-4		
0x40	TC Channel 1	See Table 33-4		
0x80	TC Channel 2	See Table 33-4		
0xC0	TC Block Control Register	TC_BCR	Write-only	–
0xC4	TC Block Mode Register	TC_BMR	Read/Write	0

TC_BCR (Block Control Register) and TC_BMR (Block Mode Register) control the whole TC block. TC channels are controlled by the registers listed in Table 33-4. The offset of each of the channel registers in Table 33-4 is in relation to the offset of the corresponding channel as mentioned in Table 33-4.

33.6.2 Channel Memory Mapping

Table 33-4. Timer/Counter (TC) Channel Memory Mapping

Offset	Register	Name	Access	Reset Value
0x00	Channel Control Register	TC_CCR	Write-only	–
0x04	Channel Mode Register	TC_CMR	Read/Write	0
0x08	Reserved	–	–	–
0x0C	Reserved	–	–	–
0x10	Counter Value	TC_CV	Read-only	0
0x14	Register A	TC_RA	Read/Write(1)	0
0x18	Register B	TC_RB	Read/Write(1)	0
0x1C	Register C	TC_RC	Read/Write	0
0x20	Status Register	TC_SR	Read-only	0
0x24	Interrupt Enable Register	TC_IER	Write-only	–
0x28	Interrupt Disable Register	TC_IDR	Write-only	–
0x2C	Interrupt Mask Register	TC_IMR	Read-only	0
0x30-0xFC	Reserved	–	–	–

Note: 1. Read only if WAVE = 0

33.6.3 TC Block Control Register

Register Name: TC_BCR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SYNC

- **SYNC: Synchro Command**

0 = No effect.

1 = Asserts the SYNC signal which generates a software trigger simultaneously for each of the channels.

33.6.4 TC Block Mode Register

Register Name: TC_BMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	TC2XC2S		TCXC1S		TC0XC0S	

- **TC0XC0S: External Clock Signal 0 Selection**

TC0XC0S		Signal Connected to XC0
0	0	TCLK0
0	1	none
1	0	TIOA1
1	1	TIOA2

- **TC1XC1S: External Clock Signal 1 Selection**

TC1XC1S		Signal Connected to XC1
0	0	TCLK1
0	1	none
1	0	TIOA0
1	1	TIOA2

- **TC2XC2S: External Clock Signal 2 Selection**

TC2XC2S		Signal Connected to XC2
0	0	TCLK2
0	1	none
1	0	TIOA0
1	1	TIOA1

33.6.5 TC Channel Control Register

Register Name: TC_CCR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	SWTRG	CLKDIS	CLKEN

- **CLKEN: Counter Clock Enable Command**

0 = No effect.

1 = Enables the clock if CLKDIS is not 1.

- **CLKDIS: Counter Clock Disable Command**

0 = No effect.

1 = Disables the clock.

- **SWTRG: Software Trigger Command**

0 = No effect.

1 = A software trigger is performed: the counter is reset and the clock is started.

33.6.6 TC Channel Mode Register: Capture Mode

Register Name: TC_CMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	LDRB		LDRA	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
WAVE = 0	CPCTRG	–	–	–	ABETRG	ETRGEDG	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LDBDIS	LDBSTOP	BURST		CLKI	TCCLKS		

- **TCCLKS: Clock Selection**

TCCLKS			Clock Selected
0	0	0	TIMER_CLOCK1
0	0	1	TIMER_CLOCK2
0	1	0	TIMER_CLOCK3
0	1	1	TIMER_CLOCK4
1	0	0	TIMER_CLOCK5
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI: Clock Invert**

0 = Counter is incremented on rising edge of the clock.

1 = Counter is incremented on falling edge of the clock.

- **BURST: Burst Signal Selection**

BURST		
0	0	The clock is not gated by an external signal.
0	1	XC0 is ANDed with the selected clock.
1	0	XC1 is ANDed with the selected clock.
1	1	XC2 is ANDed with the selected clock.

- **LDBSTOP: Counter Clock Stopped with RB Loading**

0 = Counter clock is not stopped when RB loading occurs.

1 = Counter clock is stopped when RB loading occurs.

- **LDBDIS: Counter Clock Disable with RB Loading**

0 = Counter clock is not disabled when RB loading occurs.

1 = Counter clock is disabled when RB loading occurs.

- **ETRGEDG: External Trigger Edge Selection**

ETRGEDG		Edge
0	0	none
0	1	rising edge
1	0	falling edge
1	1	each edge

- **ABETRG: TIOA or TIOB External Trigger Selection**

0 = TIOB is used as an external trigger.

1 = TIOA is used as an external trigger.

- **CPCTRG: RC Compare Trigger Enable**

0 = RC Compare has no effect on the counter and its clock.

1 = RC Compare resets the counter and starts the counter clock.

- **WAVE**

0 = Capture Mode is enabled.

1 = Capture Mode is disabled (Waveform Mode is enabled).

- **LDRA: RA Loading Selection**

LDRA		Edge
0	0	none
0	1	rising edge of TIOA
1	0	falling edge of TIOA
1	1	each edge of TIOA

- **LDRB: RB Loading Selection**

LDRB		Edge
0	0	none
0	1	rising edge of TIOA
1	0	falling edge of TIOA
1	1	each edge of TIOA

33.6.7 TC Channel Mode Register: Waveform Mode

Register Name: TC_CMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
BSWTRG		BEEVT		BCPC		BCPB	

23	22	21	20	19	18	17	16
ASWTRG		AEEVT		ACPC		ACPA	

15	14	13	12	11	10	9	8
WAVE = 1	WAVSEL		ENETRG	EEVT		EEVTEDG	

7	6	5	4	3	2	1	0
CPCDIS	CPCSTOP	BURST		CLKI	TCCLKS		

- **TCCLKS: Clock Selection**

TCCLKS			Clock Selected
0	0	0	TIMER_CLOCK1
0	0	1	TIMER_CLOCK2
0	1	0	TIMER_CLOCK3
0	1	1	TIMER_CLOCK4
1	0	0	TIMER_CLOCK5
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI: Clock Invert**

0 = Counter is incremented on rising edge of the clock.

1 = Counter is incremented on falling edge of the clock.

- **BURST: Burst Signal Selection**

BURST		
0	0	The clock is not gated by an external signal.
0	1	XC0 is ANDed with the selected clock.
1	0	XC1 is ANDed with the selected clock.
1	1	XC2 is ANDed with the selected clock.

- **CPCSTOP: Counter Clock Stopped with RC Compare**

0 = Counter clock is not stopped when counter reaches RC.

1 = Counter clock is stopped when counter reaches RC.

- **CPCDIS: Counter Clock Disable with RC Compare**

0 = Counter clock is not disabled when counter reaches RC.

1 = Counter clock is disabled when counter reaches RC.

- **EEVTEDG: External Event Edge Selection**

EEVTEDG		Edge
0	0	none
0	1	rising edge
1	0	falling edge
1	1	each edge

- **EEVT: External Event Selection**

EEVT		Signal selected as external event	TIOB Direction
0	0	TIOB	input[(1)]
0	1	XC0	output
1	0	XC1	output
1	1	XC2	output

Note: 1. *If TIOB is chosen as the external event signal, it is configured as an input and no longer generates waveforms.*

- **ENETRG: External Event Trigger Enable**

0 = The external event has no effect on the counter and its clock. In this case, the selected external event only controls the TIOA output.

1 = The external event resets the counter and starts the counter clock.

- **WAVSEL: Waveform Selection**

WAVSEL		Effect
0	0	UP mode without automatic trigger on RC Compare
1	0	UP mode with automatic trigger on RC Compare
0	1	UPDOWN mode without automatic trigger on RC Compare
1	1	UPDOWN mode with automatic trigger on RC Compare

- **WAVE = 1**

0 = Waveform Mode is disabled (Capture Mode is enabled).

1 = Waveform Mode is enabled.

- **ACPA: RA Compare Effect on TIOA**

ACPA		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle



- **ACPC: RC Compare Effect on TIOA**

ACPC		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **AEEVT: External Event Effect on TIOA**

AEEVT		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **ASWTRG: Software Trigger Effect on TIOA**

ASWTRG		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **BCPB: RB Compare Effect on TIOB**

BCPB		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **BCPC: RC Compare Effect on TIOB**

BCPC		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **BEEVT: External Event Effect on TIOB**

BEEVT		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **BSWTRG: Software Trigger Effect on TIOB**

BSWTRG		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

33.6.8 TC Counter Value Register

Register Name: TC_CV

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
CV							

7	6	5	4	3	2	1	0
CV							

- **CV: Counter Value**

CV contains the counter value in real time.

33.6.9 TC Register A

Register Name: TC_RA

Access Type: Read-only if WAVE = 0, Read/Write if WAVE = 1

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RA							

- **RA: Register A**

RA contains the Register A value in real time.

33.6.10 TC Register B

Register Name: TC_RB

Access Type: Read-only if WAVE = 0, Read/Write if WAVE = 1

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RB							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RB							

- **RB: Register B**

RB contains the Register B value in real time.

33.6.11 TC Register C

Register Name: TC_RC

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
RC							

7	6	5	4	3	2	1	0
RC							

- **RC: Register C**

RC contains the Register C value in real time.

33.6.12 TC Status Register

Register Name: TC_SR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	MTIOB	MTIOA	CLKSTA
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: Counter Overflow Status**

0 = No counter overflow has occurred since the last read of the Status Register.

1 = A counter overflow has occurred since the last read of the Status Register.

- **LOVRS: Load Overrun Status**

0 = Load overrun has not occurred since the last read of the Status Register or WAVE = 1.

1 = RA or RB have been loaded at least twice without any read of the corresponding register since the last read of the Status Register, if WAVE = 0.

- **CPAS: RA Compare Status**

0 = RA Compare has not occurred since the last read of the Status Register or WAVE = 0.

1 = RA Compare has occurred since the last read of the Status Register, if WAVE = 1.

- **CPBS: RB Compare Status**

0 = RB Compare has not occurred since the last read of the Status Register or WAVE = 0.

1 = RB Compare has occurred since the last read of the Status Register, if WAVE = 1.

- **CPCS: RC Compare Status**

0 = RC Compare has not occurred since the last read of the Status Register.

1 = RC Compare has occurred since the last read of the Status Register.

- **LDRAS: RA Loading Status**

0 = RA Load has not occurred since the last read of the Status Register or WAVE = 1.

1 = RA Load has occurred since the last read of the Status Register, if WAVE = 0.

- **LDRBS: RB Loading Status**

0 = RB Load has not occurred since the last read of the Status Register or WAVE = 1.

1 = RB Load has occurred since the last read of the Status Register, if WAVE = 0.

- **ETRGS: External Trigger Status**

0 = External trigger has not occurred since the last read of the Status Register.

1 = External trigger has occurred since the last read of the Status Register.

- **CLKSTA: Clock Enabling Status**

0 = Clock is disabled.

1 = Clock is enabled.

- **MTIOA: TIOA Mirror**

0 = TIOA is low. If WAVE = 0, this means that TIOA pin is low. If WAVE = 1, this means that TIOA is driven low.

1 = TIOA is high. If WAVE = 0, this means that TIOA pin is high. If WAVE = 1, this means that TIOA is driven high.

- **MTIOB: TIOB Mirror**

0 = TIOB is low. If WAVE = 0, this means that TIOB pin is low. If WAVE = 1, this means that TIOB is driven low.

1 = TIOB is high. If WAVE = 0, this means that TIOB pin is high. If WAVE = 1, this means that TIOB is driven high.

33.6.13 TC Interrupt Enable Register

Register Name: TC_IER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: Counter Overflow**

0 = No effect.

1 = Enables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun**

0 = No effect.

1 = Enables the Load Overrun Interrupt.

- **CPAS: RA Compare**

0 = No effect.

1 = Enables the RA Compare Interrupt.

- **CPBS: RB Compare**

0 = No effect.

1 = Enables the RB Compare Interrupt.

- **CPCS: RC Compare**

0 = No effect.

1 = Enables the RC Compare Interrupt.

- **LDRAS: RA Loading**

0 = No effect.

1 = Enables the RA Load Interrupt.

- **LDRBS: RB Loading**

0 = No effect.

1 = Enables the RB Load Interrupt.

- **ETRGS: External Trigger**

0 = No effect.

1 = Enables the External Trigger Interrupt.



33.6.14 TC Interrupt Disable Register

Register Name: TC_IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: Counter Overflow**

0 = No effect.

1 = Disables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun**

0 = No effect.

1 = Disables the Load Overrun Interrupt (if WAVE = 0).

- **CPAS: RA Compare**

0 = No effect.

1 = Disables the RA Compare Interrupt (if WAVE = 1).

- **CPBS: RB Compare**

0 = No effect.

1 = Disables the RB Compare Interrupt (if WAVE = 1).

- **CPCS: RC Compare**

0 = No effect.

1 = Disables the RC Compare Interrupt.

- **LDRAS: RA Loading**

0 = No effect.

1 = Disables the RA Load Interrupt (if WAVE = 0).

- **LDRBS: RB Loading**

0 = No effect.

1 = Disables the RB Load Interrupt (if WAVE = 0).

- **ETRGS: External Trigger**

0 = No effect.

1 = Disables the External Trigger Interrupt.

33.6.15 TC Interrupt Mask Register

Register Name: TC_IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: Counter Overflow**

0 = The Counter Overflow Interrupt is disabled.

1 = The Counter Overflow Interrupt is enabled.

- **LOVRS: Load Overrun**

0 = The Load Overrun Interrupt is disabled.

1 = The Load Overrun Interrupt is enabled.

- **CPAS: RA Compare**

0 = The RA Compare Interrupt is disabled.

1 = The RA Compare Interrupt is enabled.

- **CPBS: RB Compare**

0 = The RB Compare Interrupt is disabled.

1 = The RB Compare Interrupt is enabled.

- **CPCS: RC Compare**

0 = The RC Compare Interrupt is disabled.

1 = The RC Compare Interrupt is enabled.

- **LDRAS: RA Loading**

0 = The Load RA Interrupt is disabled.

1 = The Load RA Interrupt is enabled.

- **LDRBS: RB Loading**

0 = The Load RB Interrupt is disabled.

1 = The Load RB Interrupt is enabled.

- **ETRGS: External Trigger**

0 = The External Trigger Interrupt is disabled.

1 = The External Trigger Interrupt is enabled.



34. MultiMedia Card Interface (MCI)

34.1 Overview

The MultiMedia Card Interface (MCI) supports the MultiMediaCard (MMC) Specification V2.2 and the SD Memory Card Specification V1.0.

The MCI includes a command register, response registers, data registers, timeout counters and error detection logic that automatically handle the transmission of commands and, when required, the reception of the associated responses and data with a limited processor overhead.

The MCI supports stream, block and multi-block data read and write, and is compatible with the Peripheral Data Controller channels, minimizing processor intervention for large buffer transfers.

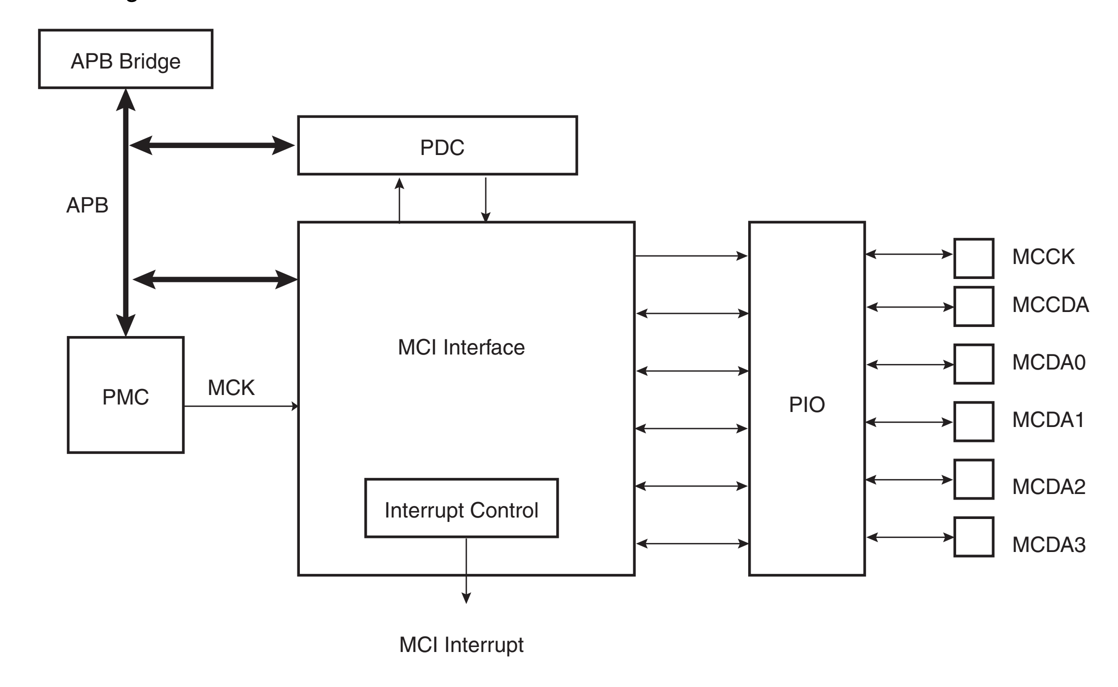
The MCI operates at a rate of up to Master Clock divided by 2 and supports the interfacing of one slot(s). Each slot may be used to interface with a MultiMediaCard bus (up to 30 Cards) or with a SD Memory Card. Only one slot can be selected at a time (slots are multiplexed). A bit field in the SD Card Register performs this selection.

The SD Memory Card communication is based on a 9-pin interface (clock, command, four data and three power lines) and the MultiMediaCard on a 7-pin interface (clock, command, one data, three power lines and one reserved for future use).

The SD Memory Card interface also supports MultiMedia Card operations. The main differences between SD and MultiMedia Cards are the initialization process and the bus topology.

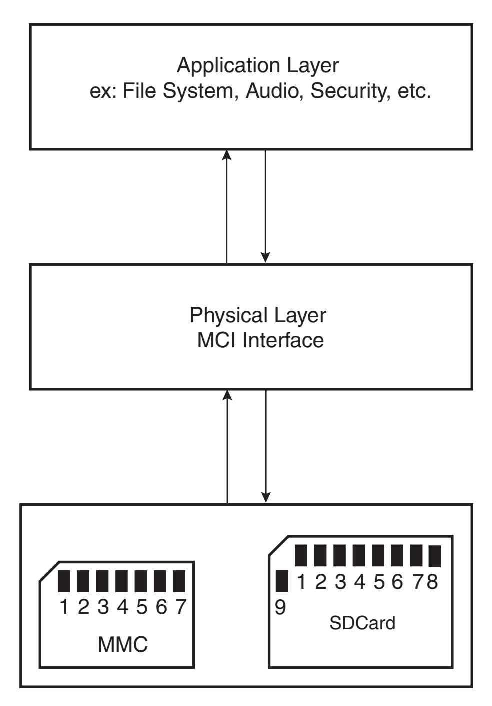
34.2 Block Diagram

Figure 34-1. Block Diagram



34.3 Application Block Diagram

Figure 34-2. Application Block Diagram



34.4 Pin Name List

Table 34-1. I/O Lines Description

Pin Name	Pin Description	Type(1)	Comments
MCCDA	Command/response	I/O/PP/OD	CMD of an MMC or SD Card
MCCK	Clock	I/O	CLK of an MMC or SD Card
MCDA0 - MCDA3	Data 0..3 of Slot A	I/O/PP	DAT0 of an MMC DAT[0..3] of an SD Card

Note: 1. I: Input, O: Output, PP: Push/Pull, OD: Open Drain.

34.5 Product Dependencies

34.5.1 I/O Lines

The pins used for interfacing the MultiMedia Cards or SD Cards may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the peripheral functions to MCI pins.

34.5.2 Power Management

The MCI may be clocked through the Power Management Controller (PMC), so the programmer must first to configure the PMC to enable the MCI clock.

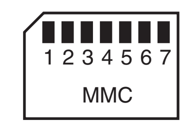
34.5.3 Interrupt

The MCI interface has an interrupt line connected to the Advanced Interrupt Controller (AIC).

Handling the MCI interrupt requires programming the AIC before configuring the MCI.

34.6 Bus Topology

Figure 34-3. Multimedia Memory Card Bus Topology



The MultiMedia Card communication is based on a 7-pin serial bus interface. It has three communication lines and four supply lines.

Table 34-2. Bus Topology

Pin Number	Name	Type[(1)]	Description	MCI Pin Name (Slot x)
1	RSV	NC	Not connected	
2	CMD	I/O/PP/OD	Command/response	MCCDx
3	VSS1	S	Supply voltage ground	VSS
4	VDD	S	Supply voltage	VDD
5	CLK	I/O	Clock	MCCK
6	VSS2	S	Supply voltage ground	VSS
7	DAT[0]	I/O/PP	Data 0	MCDx0

Note: 1. I: Input, O: Output, PP: Push/Pull, OD: Open Drain.

Figure 34-4. MMC Bus Connections (One Slot)

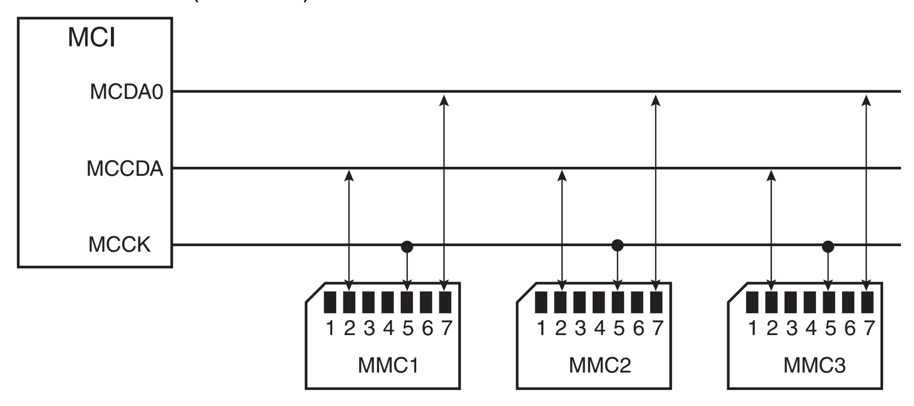
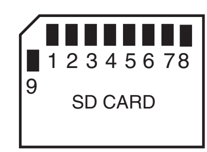


Figure 34-5. SD Memory Card Bus Topology



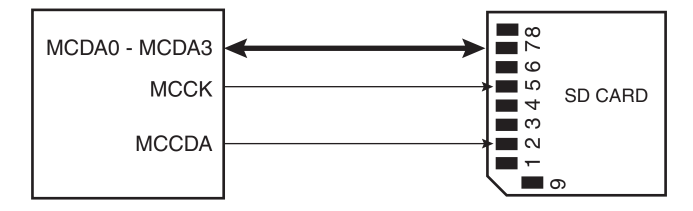
The SD Memory Card bus includes the signals listed in Table 34-3.

Table 34-3. SD Memory Card Bus Signals

Pin Number	Name	Type[(1)]	Description	MCI Pin Name (Slot x)
1	CD/DAT[3]	I/O/PP	Card detect/ Data line Bit 3	MCDx3
2	CMD	PP	Command/response	MCCDx
3	VSS1	S	Supply voltage ground	VSS
4	VDD	S	Supply voltage	VDD
5	CLK	I/O	Clock	MCCK
6	VSS2	S	Supply voltage ground	VSS
7	DAT[0]	I/O/PP	Data line Bit 0	MCDx0
8	DAT[1]	I/O/PP	Data line Bit 1	MCDx1
9	DAT[2]	I/O/PP	Data line Bit 2	MCDx2

Note: 1. I: input, O: output, PP: Push Pull, OD: Open Drain

Figure 34-6. SD Card Bus Connections with One Slot



When the MCI is configured to operate with SD memory cards, the width of the data bus can be selected in the MCI_SDCR register. Clearing the SDCBUS bit in this register means that



the width is one bit; setting it means that the width is four bits. In the case of multimedia cards, only the data line 0 is used. The other data lines can be used as independent PIOs.

34.7 MultiMedia Card Operations

After a power-on reset, the cards are initialized by a special message-based MultiMedia Card bus protocol. Each message is represented by one of the following tokens:

- Command: A command is a token that starts an operation. A command is sent from the host either to a single card (addressed command) or to all connected cards (broadcast command). A command is transferred serially on the CMD line.
- Response: A response is a token which is sent from an addressed card or (synchronously) from all connected cards to the host as an answer to a previously received command. A response is transferred serially on the CMD line.
- Data: Data can be transferred from the card to the host or vice versa. Data is transferred via the data line.

Card addressing is implemented using a session address assigned during the initialization phase by the bus controller to all currently connected cards. Their unique CID number identifies individual cards.

The structure of commands, responses and data blocks is described in the MultiMedia-Card System Specification Version 2.2. See also Table 34-4 on page 500.

MultiMediaCard bus data transfers are composed of these tokens.

There are different types of operations. Addressed operations always contain a command and a response token. In addition, some operations have a data token; the others transfer their information directly within the command or response structure. In this case, no data token is present in an operation. The bits on the DAT and the CMD lines are transferred synchronous to the clock MCCK.

Two types of data transfer commands are defined:

- Sequential commands: These commands initiate a continuous data stream. They are terminated only when a stop command follows on the CMD line. This mode reduces the command overhead to an absolute minimum.
- Block-oriented commands: These commands send a data block succeeded by CRC bits.

Both read and write operations allow either single or multiple block transmission. A multiple block transmission is terminated when a stop command follows on the CMD line similarly to the sequential read.

The MCI provides a set of registers to perform the entire range of MultiMedia Card operations.

34.7.1 Command - Response Operation

After reset, the MCI is disabled and becomes valid after setting the MCIEN bit in the MCI_CR Control Register. The bit PWSEN saves power by dividing the MCI clock by $2^{PWSDIV} + 1$ when the bus is inactive.

The command and the response of the card are clocked out with the rising edge of the MCCK.

All the timings for MultiMedia Card are defined in the MultiMediaCard System Specification Version 2.2.

The two bus modes (open drain and push/pull) needed to process all the operations are defined in the MCI command register. The MCI_CMDR allows a command to be carried out.

For example, to perform an ALL_SEND_CID command:

	Host Command					N_{ID} Cycles					CID			
CMD	S	T	Content	CRC	E	Z	******	Z	S	T	Content	Z	Z	Z

The command ALL_SEND_CID and the fields and values for the MCI_CMDR Control Register are described in Table 34-4 and Table 34-5.

Table 34-4. ALL_SEND_CID Command Description

CMD Index	Type	Argument	Resp	Abbreviation	Command Description
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line

Note: bcr means broadcast command with response.

Table 34-5. Fields and Values for MCI_CMDR Command Register

Field	Value
CMDNB (command number)	2 (CMD2)
RSPTYP (response type)	2 (R2: 136 bits response)
SPCMD (special command)	0 (not a special command)
OPCMD (open drain command)	1
MAXLAT (max latency for command to response)	0 (NID cycles ==> 5 cycles)
TRCMD (transfer command)	0 (No transfer)
TRDIR (transfer direction)	X (available only in transfer command)
TRTYP (transfer type)	X (available only in transfer command)

The MCI_ARGR contains the argument field of the command.

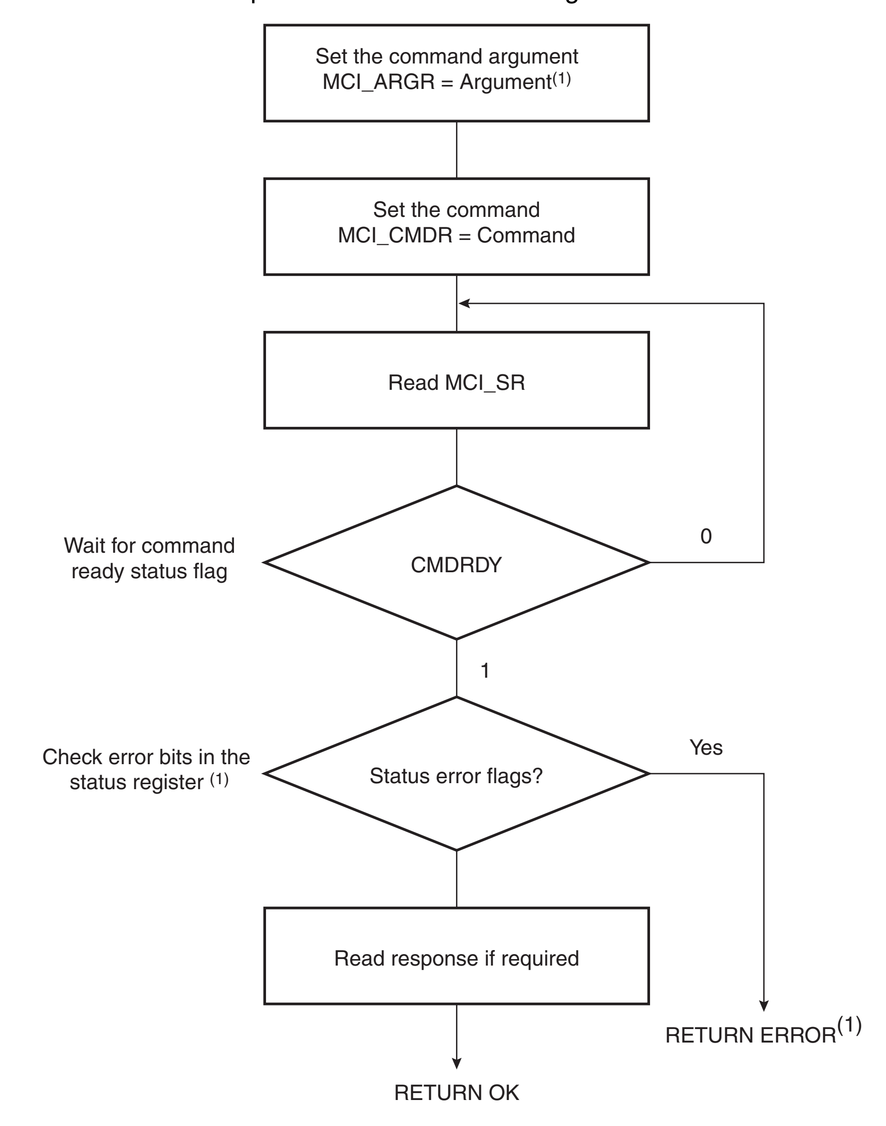
To send a command, the user must perform the following steps:

- Fill the argument register (MCI_ARGR) with the command argument.
- Set the command register (MCI_CMDR) (see Table 34-5).

The command is sent immediately after writing the command register. The status bit CMDRDY in the status register (MCI_SR) is asserted when the command is completed. If the command requires a response, it can be read in the MCI response register (MCI_RSPR). The response size can be from 48 bits up to 136 bits depending on the command. The MCI embeds an error detection to prevent any corrupted data during the transfer.

The following flowchart shows how to send a command to the card and read the response if needed. In this example, the status register bits are polled but setting the appropriate bits in the interrupt enable register (MCI_IER) allows using an interrupt method.

Figure 34-7. Command/Response Functional Flow Diagram



Note: 1. If the command is SEND_OP_COND, the CRC error flag is always present (refer to R3 response in the MultiMediaCard specification).

34.7.2 Data Transfer Operation

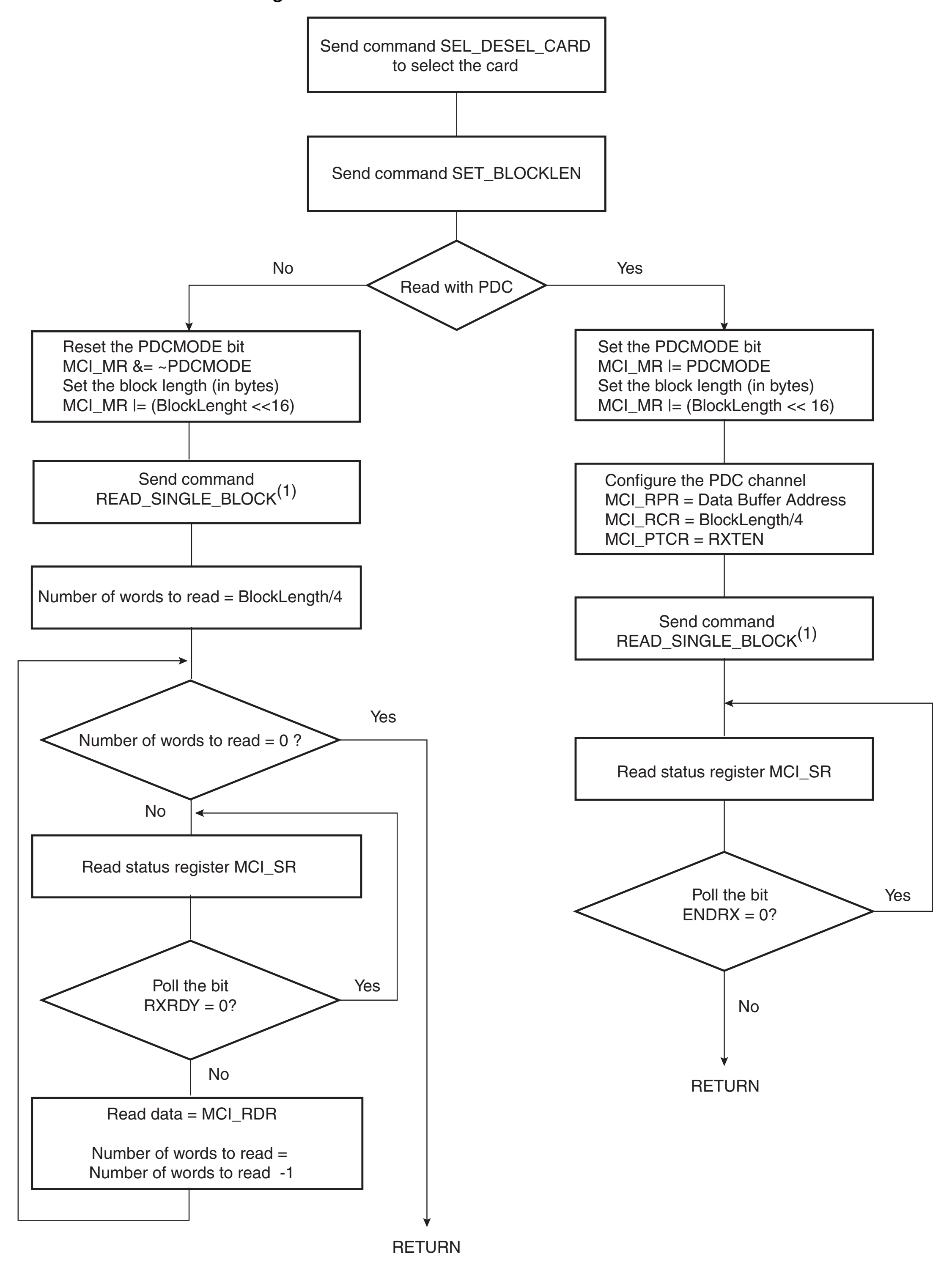
The MultiMedia Card allows several read/write operations (single block, multiple blocks, stream, etc.).

These operations can be done using the features of the Peripheral Data Controller (PDC). If the PDCMODE bit is set in MCI_MR, then all reads and writes use the PDC facilities. In all cases, the block length must be defined in the mode register.

34.7.3 Read Operation

The following flowchart shows how to read a single block with or without use of PDC facilities. In this example (see Figure 34-8), a polling method is used to wait for the end of read. Similarly, the user can configure the interrupt enable register (MCI_IER) to trigger an interrupt at the end of read. These two methods can be applied for all MultiMedia Card read functions.

Figure 34-8. Read Functional Flow Diagram



Note: 1. This command is supposed to have been correctly sent (see Figure 34-7).

34.7.4 Write Operation

In write operation, the MCI Mode Register (MCI_MR) is used to define the padding value when writing non-multiple block size. If the bit PDCPADV is 0, then 0x00 value is used when padding data, otherwise 0xFF is used.

If set, the bit PDCMODE enables PDC transfer.

The following flowchart shows how to write a single block with or without use of PDC facilities (see Figure 34-9).

Polling or interrupt method can be used to wait for the end of write according to the contents of the Interrupt Mask Register (MCI_IMR).

This flowchart can be adapted to perform all the MultiMedia Card write functions.

Figure 34-9. Write Functional Flow Diagram

Send command SEL_DESEL_CARD to select the card

Send command SET_BLOCKLEN

Write using PDC

No

Reset the PDCMODE bit
MCI_MR &= ~PDCMODE
Set the block length
MCI_MR |= (BlockLenght <<16)

Send command WRITE_SINGLE_BLOCK(1)

Number of words to write = BlockLength/4

Number of words to write = 0 ?

Yes

RETURN

No

Read status register MCI_SR

Poll the bit TXRDY = 0?

Yes

No

MCI_TDR = Data to write

Number of words to write = Number of words to write -1

Yes

Set the PDCMODE bit
MCI_MR |= PDCMODE
Set the block length
MCI_MR |= (BlockLength << 16)

Configure the PDC channel
MCI_TPR = Data Buffer Address to write
MCI_TCR = BlockLength/4

Send command WRITE_SINGLE_BLOCK(1)

MCI_PTCR = TXTEN

Read status register MCI_SR

Poll the bit ENDTX = 0?

Yes

No

RETURN

Note: 1. This command is supposed to have been correctly sent (see Figure 34-7).

34.8 SD Card Operations

The MultiMedia Card Interface allows processing of SD Memory Card (Secure Digital Memory Card) commands. The SD Memory Card includes a copyright protection mechanism that complies with the security requirements of the SDMI standard (Secure Digital Music Initiative), is faster and applicable to higher memory capacity.

The physical form factor, pin assignment and data transfer protocol are forward-com-patible with the MultiMedia Card with some additions.

The SD Memory Card communication is based on a 9-pin interface (Clock, Command, 4 x Data and 3 x Power lines). The communication protocol is defined as a part of this specification. The main difference between the SD Memory Card and the MultiMedia Card is the initialization process.

The SD Card Register (MCI_SDCR) allows selection of the Card Slot and the data bus width.

The SD Card bus allows dynamic configuration of the number of data lines. After power up, by default, the SD Memory Card uses only DAT0 for data transfer. After initialization, the host can change the bus width (number of active data lines).

34.9 MultiMedia Card Interface (MCI) User Interface

Table 34-6. MCI Register Mapping

Offset	Register	Register Name	Read/Write	Reset
0x00	Control Register	MCI_CR	Write	–
0x04	Mode Register	MCI_MR	Read/write	0x0
0x08	Data Timeout Register	MCI_DTOR	Read/write	0x0
0x0C	SD Card Register	MCI_SDCR	Read/write	0x0
0x10	Argument Register	MCI_ARGR	Read/write	0x0
0x14	Command Register	MCI_CMDR	Write	–
0x18 - 0x1C	Reserved	–	–	–
0x20	Response Register(1)	MCI_RSPR	Read	0x0
0x24	Response Register(1)	MCI_RSPR	Read	0x0
0x28	Response Register(1)	MCI_RSPR	Read	0x0
0x2C	Response Register(1)	MCI_RSPR	Read	0x0
0x30	Receive Data Register	MCI_RDR	Read	0x0
0x34	Transmit Data Register	MCI_TDR	Write	–
0x38 - 0x3C	Reserved	–	–	–
0x40	Status Register	MCI_SR	Read	0xC0E5
0x44	Interrupt Enable Register	MCI_IER	Write	–
0x48	Interrupt Disable Register	MCI_IDR	Write	–
0x4C	Interrupt Mask Register	MCI_IMR	Read	0x0
0x50-0xFC	Reserved	–	–	–
0x100-0x124	Reserved for the PDC	–	–	–

Note: 1. The response register can be read by N accesses at the same MCI_RSPR or at consecutive addresses (0x20 to 0x2C). N depends on the size of the response.



34.9.1 MCI Control Register

Name: MCI_CR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SWRST	–	–	–	PWSDIS	PWSEN	MCIDIS	MCIEN

- **MCIEN: Multi-Media Interface Enable**

0 = No effect.

1 = Enables the Multi-Media Interface if MCDIS is 0.

- **MCIDIS: Multi-Media Interface Disable**

0 = No effect.

1 = Disables the Multi-Media Interface.

- **PWSEN: Power Save Mode Enable**

0 = No effect.

1 = Enables the Power Saving Mode if PWSDIS is 0.

Warning: Before enabling this mode, the user must set a value different from 0 in the PWSDIV field (Mode Register MCI_MR) .

- **PWSDIS: Power Save Mode Disable**

0 = No effect.

1 = Disables the Power Saving Mode.

- **SWRST: Software Reset**

0 = No effect.

1 = Resets the MCI. A software triggered hardware reset of the MCI interface is performed.



34.9.2 MCI Mode Register

Name: MCI_MR

Access Type: Read/write

31	30	29	28	27	26	25	24
–	–	BLKLEN					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
BLKLEN						0	0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PDCMODE	PDCPADV	–	–	–	PWSDIV		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CLKDIV							

- **CLKDIV: Clock Divider**

Multimedia Card Interface clock (MCCK) is Master Clock (MCK) divided by (2*(CLKDIV+1)).

- **PWSDIV: Power Saving Divider**

Multimedia Card Interface clock is divided by $2^{(PWSDIV)}$ + 1 when entering Power Saving Mode.

<u>**Warning:**</u> This value must be different from 0 before enabling the Power Save Mode in the MCI_CR (MCI_PWSEN bit).

- **PDCPADV: PDC Padding Value**

0 = 0x00 value is used when padding data in write transfer (not only PDC transfer).

1 = 0xFF value is used when padding data in write transfer (not only PDC transfer).

- **PDCMODE: PDC-oriented Mode**

0 = Disables PDC transfer

1 = Enables PDC transfer. In this case, UNRE and OVRE flags in the MCI Mode Register (MCI_SR) are deactivated after the PDC transfer has been completed.

- **BLKLEN: Data Block Length**

This field determines the size of the data block.

Bits 16 and 17 must be 0.

34.9.3 MCI Data Timeout Register

Name: MCI_DTOR

Access Type: Read/write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	DTOMUL			DTOCYC			

- **DTOCYC: Data Timeout Cycle Number**
- **DTOMUL: Data Timeout Multiplier**

These fields determine the maximum number of Master Clock cycles that the MCI waits between two data block transfers. It equals (DTOCYC x Multiplier).

Multiplier is defined by DTOMUL as shown in the following table:

DTOMUL			Multiplier
0	0	0	1
0	0	1	16
0	1	0	128
0	1	1	256
1	0	0	1024
1	0	1	4096
1	1	0	65536
1	1	1	1048576

If the data time-out set by DTOCYC and DTOMUL has been exceeded, the Data Time-out Error flag (DTOE) in the MCI Status Register (MCI_SR) raises.

34.9.4 MCI SD Card Register

Name: MCI_SDCR

Access Type: Read/write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
SDCBUS	–	–	–	SDCSEL			

- **SDCSEL: SD Card Selector**

0 = SDCARD Slot A selected.

- **SDCBUS: SD Card Bus Width**

0 = 1-bit data bus

1 = 4-bit data bus

34.9.5 MCI Argument Register

Name: MCI_ARGR

Access Type: Read/write

31	30	29	28	27	26	25	24
ARG							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ARG							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ARG							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ARG							

- **ARG: Command Argument**



34.9.6 MCI Command Register

Name: MCI_CMDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	TRTYP		TRDIR	TRCMD	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	MAXLAT	OPDCMD	SPCMD		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RSPTYP		CMDNB					

This register is write-protected while CMDRDY is 0 in MCI_SR. If an Interrupt command is sent, this register is only write-able by an interrupt response (field SPCMD). This means that the current command execution cannot be interrupted or modified.

- **CMDNB: Command Number**
- **RSPTYP: Response Type**

RSP		Response Type
0	0	No response.
0	1	48-bit response.
1	0	136-bit response.
1	1	Reserved.

- **SPCMD: Special Command**

SPCMD			Command
0	0	0	Not a special CMD.
0	0	1	Initialization CMD: 74 clock cycles for initialization sequence.
0	1	0	Synchronized CMD: Wait for the end of the current data block transfer before sending the pending command.
0	1	1	Reserved.
1	0	0	Interrupt command: Corresponds to the Interrupt Mode (CMD40).
1	0	1	Interrupt response: Corresponds to the Interrupt Mode (CMD40).

- **OPDCMD: Open Drain Command**

0 = Push pull command

1 = Open drain command



- **MAXLAT: Max Latency for Command to Response**

0 = 5-cycle max latency

1 = 64-cycle max latency

- **TRCMD: Transfer Command**

TRCMD		Transfer Type
0	0	No data transfer
0	1	Start data transfer
1	0	Stop data transfer
1	1	Reserved

- **TRDIR: Transfer Direction**

0 = Write

1 = Read

- **TRTYP: Transfer Type**

TRTYP		Transfer Type
0	0	Block
0	1	Multiple Block
1	0	Stream
1	1	Reserved

34.9.7 MCI SD Response Register

Name: MCI_RSPR

Access Type: Read-only

31	30	29	28	27	26	25	24
RSP							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RSP							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RSP							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RSP							

- **RSP: Response**

Note: 1. The response register can be read by N accesses at the same MCI_RSPR or at consecutive addresses (0x20 to 0x2C). N depends on the size of the response.



34.9.8 MCI SD Receive Data Register

Name: MCI_RDR

Access Type: Read-only

31	30	29	28	27	26	25	24
DATA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DATA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DATA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA							

- **DATA: Data to Read**

34.9.9 MCI SD Transmit Data Register

Name: MCI_TDR

Access Type: Write-only

31	30	29	28	27	26	25	24
DATA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DATA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DATA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA							

- **DATA: Data to Write**

34.9.10 MCI Status Register

Name: MCI_SR

Access Type: Read-only

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXBUFE	RXBUFF	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **CMDRDY: Command Ready**

0 = A command is in progress.

1 = The last command has been sent. Cleared when writing in the MCI_CMDR.

- **RXRDY: Receiver Ready**

0 = Data has not yet been received since the last read of MCI_RDR.

1 = Data has been received since the last read of MCI_RDR.

- **TXRDY: Transmit Ready**

0= The last data written in MCI_TDR has not yet been transferred in the Shift Register.

1= The last data written in MCI_TDR has been transferred in the Shift Register.

- **BLKE: Data Block Ended**

0 = A data block transfer is not yet finished.

1 = A data block transfer has ended. Set at the end of the last block in PDCMODE (when RXBUFF or TXBUFE is set), otherwise at the end of the first block. Cleared when reading the MCI_SR.

- **DTIP: Data Transfer in Progress**

0 = No data transfer in progress.

1 = The current data transfer is still in progress, including CRC16 calculation. Cleared at the end of the CRC16 calculation.

- **NOTBUSY: Data Not Busy**

0 = The card is not ready for new data transfer.

1 = The card is ready for new data transfer (Data line DAT0 high corresponding to a free data receive buffer in the card).

- **ENDRX: End of RX Buffer**

0 = The Receive Counter Register has not reached 0 since the last write in MCI_RCR or MCI_RNCR.

1 = The Receive Counter Register has reached 0 since the last write in MCI_RCR or MCI_RNCR.

- **ENDTX: End of TX Buffer**

0 = The Transmit Counter Register has not reached 0 since the last write in MCI_TCR or MCI_TNCR.

1 = The Transmit Counter Register has reached 0 since the last write in MCI_TCR or MCI_TNCR.

- **RXBUFF: RX Buffer Full**



0 = MCI_RCR or MCI_RNCR has a value other than 0.

1 = Both MCI_RCR and MCI_RNCR have a value of 0.

- **TXBUFE: TX Buffer Empty**

0 = MCI_TCR or MCI_TNCR has a value other than 0.

1 = Both MCI_TCR and MCI_TNCR have a value of 0.

- **RINDE: Response Index Error**

0 = No error.

1 = A mismatch is detected between the command index sent and the response index received. Cleared when writing in the MCI_CMDR.

- **RDIRE: Response Direction Error**

0 = No error.

1 = The direction bit from card to host in the response has not been detected.

- **RCRCE: Response CRC Error**

0 = No error.

1 = A CRC7 error has been detected in the response. Cleared when writing in the MCI_CMDR.

- **RENDE: Response End Bit Error**

0 = No error.

1 = The end bit of the response has not been detected. Cleared when writing in the MCI_CMDR.

- **RTOE: Response Time-out Error**

0 = No error.

1 = The response time-out set by MAXLAT in the MCI_CMDR has been exceeded. Cleared when writing in the MCI_CMDR.

- **DCRCE: Data CRC Error**

0 = No error.

1 = A CRC16 error has been detected in the last data block. Cleared when sending a new data transfer command.

- **DTOE: Data Time-out Error**

0 = No error.

1 = The data time-out set by DTOCYC and DTOMUL in MCI_DTOR has been exceeded. Cleared when writing in the MCI_CMDR.

- **OVRE: Overrun**

0 = No error.

1 = At least one 8-bit received data has been lost (not read). Cleared when sending a new data transfer command.

- **UNRE: Underrun**

0 = No error.

1 = At least one 8-bit data has been sent without valid information (not written). Cleared when sending a new data transfer command.

34.9.11 MCI Interrupt Enable Register

Name: MCI_IER

Access Type: Write-only

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXBUFE	RXBUFF	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **CMDRDY: Command Ready Interrupt Enable**
- **RXRDY: Receiver Ready Interrupt Enable**
- **TXRDY: Transmit Ready Interrupt Enable**
- **BLKE: Data Block Ended Interrupt Enable**
- **DTIP: Data Transfer in Progress Interrupt Enable**
- **NOTBUSY: Data Not Busy Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable**
- **ENDTX: End of Transmit Buffer Interrupt Enable**
- **RXBUFF: Receive Buffer Full Interrupt Enable**
- **TXBUFE: Transmit Buffer Empty Interrupt Enable**
- **RINDE: Response Index Error Interrupt Enable**
- **RDIRE: Response Direction Error Interrupt Enable**
- **RCRCE: Response CRC Error Interrupt Enable**
- **RENDE: Response End Bit Error Interrupt Enable**
- **RTOE: Response Time-out Error Interrupt Enable**
- **DCRCE: Data CRC Error Interrupt Enable**
- **DTOE: Data Time-out Error Interrupt Enable**
- **OVRE: Overrun Interrupt Enable**
- **UNRE: UnderRun Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.



34.9.12 MCI Interrupt Disable Register

Name: MCI_IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXBUFE	RXBUFF	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **CMDRDY: Command Ready Interrupt Disable**
- **RXRDY: Receiver Ready Interrupt Disable**
- **TXRDY: Transmit Ready Interrupt Disable**
- **BLKE: Data Block Ended Interrupt Disable**
- **DTIP: Data Transfer in Progress Interrupt Disable**
- **NOTBUSY: Data Not Busy Interrupt Disable**
- **ENDRX: End of Receive Buffer Interrupt Disable**
- **ENDTX: End of Transmit Buffer Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**
- **TXBUFE: Transmit Buffer Empty Interrupt Disable**
- **RINDE: Response Index Error Interrupt Disable**
- **RDIRE: Response Direction Error Interrupt Disable**
- **RCRCE: Response CRC Error Interrupt Disable**
- **RENDE: Response End Bit Error Interrupt Disable**
- **RTOE: Response Time-out Error Interrupt Disable**
- **DCRCE: Data CRC Error Interrupt Disable**
- **DTOE: Data Time-out Error Interrupt Disable**
- **OVRE: Overrun Interrupt Disable**
- **UNRE: UnderRun Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.



34.9.13 MCI Interrupt Mask Register

Name: MCI_IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXBUFE	RXBUFF	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **CMDRDY: Command Ready Interrupt Mask**
- **RXRDY: Receiver Ready Interrupt Mask**
- **TXRDY: Transmit Ready Interrupt Mask**
- **BLKE: Data Block Ended Interrupt Mask**
- **DTIP: Data Transfer in Progress Interrupt Mask**
- **NOTBUSY: Data Not Busy Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **ENDTX: End of Transmit Buffer Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**
- **TXBUFE: Transmit Buffer Empty Interrupt Mask**
- **RINDE: Response Index Error Interrupt Mask**
- **RDIRE: Response Direction Error Interrupt Mask**
- **RCRCE: Response CRC Error Interrupt Mask**
- **RENDE: Response End Bit Error Interrupt Mask**
- **RTOE: Response Time-out Error Interrupt Mask**
- **DCRCE: Data CRC Error Interrupt Mask**
- **DTOE: Data Time-out Error Interrupt Mask**
- **OVRE: Overrun Interrupt Mask**
- **UNRE: UnderRun Interrupt Mask**

0 = The corresponding interrupt is not enabled.

1 = The corresponding interrupt is enabled.



35. USB Host Port (UHP)

35.1 Description

The USB Host Port (UHP) interfaces the USB with the host application. It handles Open HCI protocol (Open Host Controller Interface) as well as USB v2.0 Full-speed and Low-speed protocols.

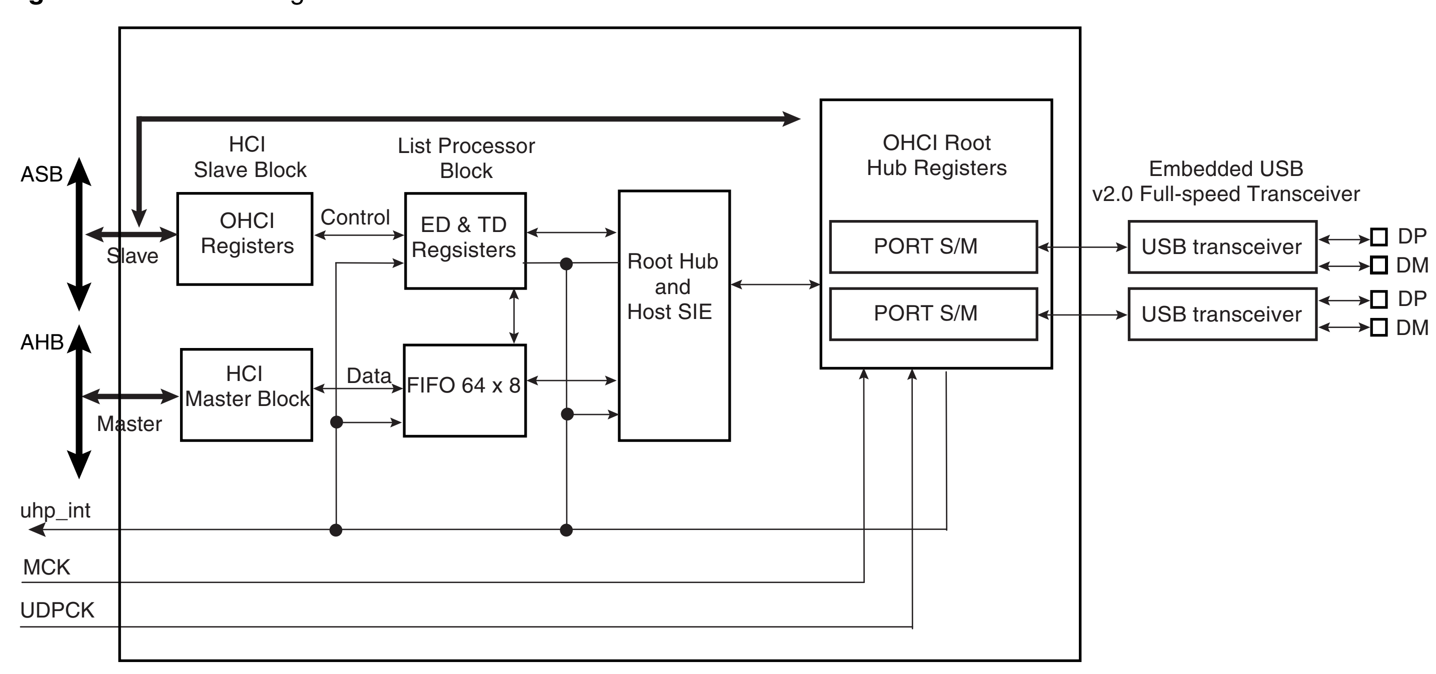
The USB Host Port integrates a root hub and transceivers on downstream ports. It provides several high-speed half-duplex serial communication ports at a baud rate of 12 Mbit/s. Up to 127 USB devices (printer, camera, mouse, keyboard, disk, etc.) and the USB hub can be connected to the USB host in the USB "tiered star" topology.

The USB Host Port controller is fully compliant with the Open HCI specification. The standard OHCI USB stack driver can be easily ported to ATMEL's architecture in the same way all existing class drivers run without hardware specialization.

This means that all standard class devices are automatically detected and available to the user application. As an example, integrating an HID (Human Interface Device) class driver provides a plug & play feature for all USB keyboards and mouses.

35.2 Block Diagram

Figure 35-1. Block Diagram



Access to the USB host operational registers is achieved through the AHB bus slave interface. The Open HCI host controller initializes master DMA transfers through the ASB bus master interface as follows:

- Fetches endpoint descriptors and transfer descriptors
- Access to endpoint data from system memory
- Access to the HC communication area
- Write status and retire transfer Descriptor



Memory access errors (abort, misalignment) lead to an “UnrecoverableError” indicated by the corresponding flag in the host controller operational registers.

The USB root hub is integrated in the USB host. Several USB downstream ports are available. The number of downstream ports can be determined by the software driver reading the root hub’s operational registers. Device connection is automatically detected by the USB host port logic.

Warning: A pull-down must be connected to DP on the board. Otherwise The USB host will permanently detect a device connection on this port.

USB physical transceivers are integrated in the product and driven by the root hub’s ports.

Over current protection on ports can be activated by the USB host controller. Atmel’s standard product does not dedicate pads to external over current protection.

35.3 Product Dependencies

35.3.1 I/O Lines

DPs and DMs are not controlled by any PIO controllers. The embedded USB physical transceivers are controlled by the USB host controller.

35.3.2 Power Management

The USB host controller requires a 48 MHz clock. This clock must be generated by a PLL with a correct accuracy of ± 0.25%.

Thus the USB device peripheral receives two clocks from the Power Management Controller (PMC): the master clock MCK used to drive the peripheral user interface (MCK domain) and the UHPCLK 48 MHz clock used to interface with the bus USB signals (Recovered 12 MHz domain).

35.3.3 Interrupt

The USB host interface has an interrupt line connected to the Advanced Interrupt Controller (AIC).

Handling USB host interrupts requires programming the AIC before configuring the UHP.

35.4 Functional Description

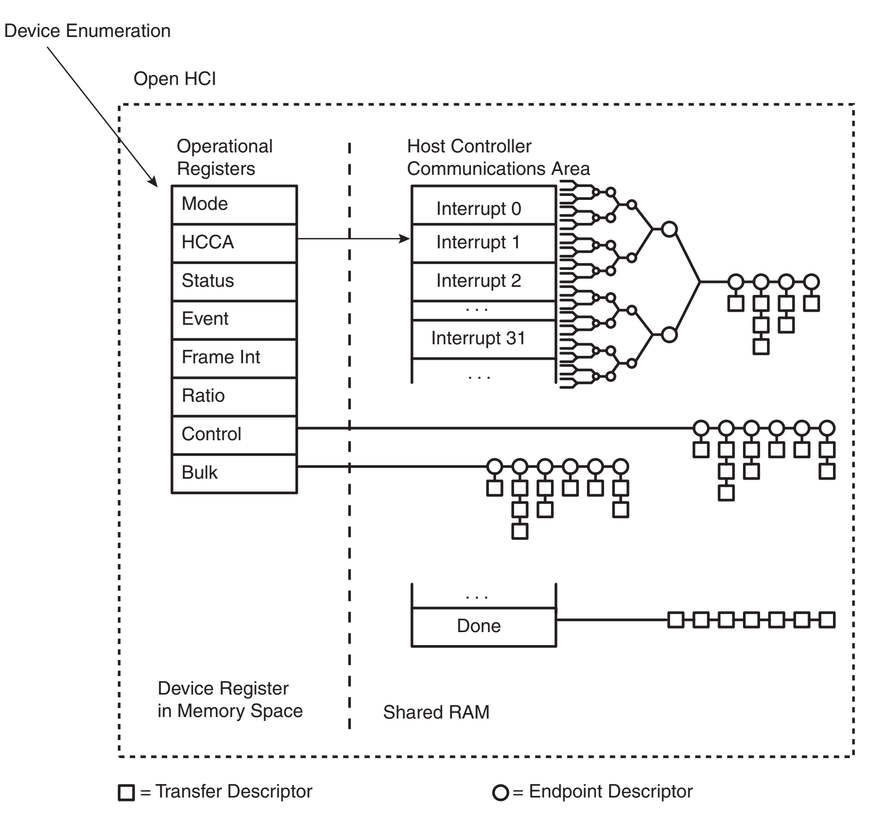
Please refer to the Open Host Controller Interface Specification for USB Release 1.0.a.

35.4.1 Host Controller Interface

There are two communication channels between the Host Controller and the Host Controller Driver. The first channel uses a set of operational registers located on the USB Host Controller. The Host Controller is the target for all communications on this channel. The operational registers contain control, status and list pointer registers. They are mapped in the memory mapped area. Within the operational register set there is a pointer to a location in the processor address space named the Host Controller Communication Area (HCCA). The HCCA is the second communication channel. The host controller is the master for all communication on this channel. The HCCA contains the head pointers to the interrupt Endpoint Descriptor lists, the head pointer to the done queue and status information associated with start-of-frame processing.

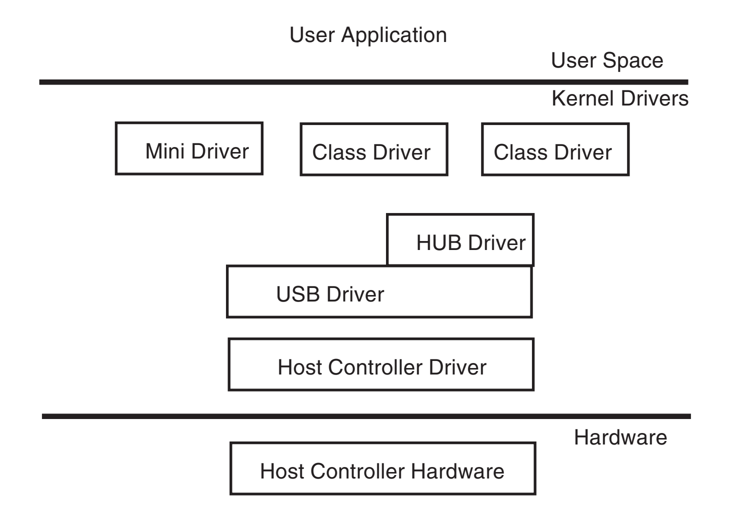
The basic building blocks for communication across the interface are Endpoint Descriptors (ED, 4 double words) and Transfer Descriptors (TD, 4 or 8 double words). The host controller assigns an Endpoint Descriptor to each endpoint in the system. A queue of Transfer Descriptors is linked to the Endpoint Descriptor for the specific endpoint.

Figure 35-2. USB Host Communication Channels



35.4.2 HOST Controller Driver

Figure 35-3. USB Host Drivers



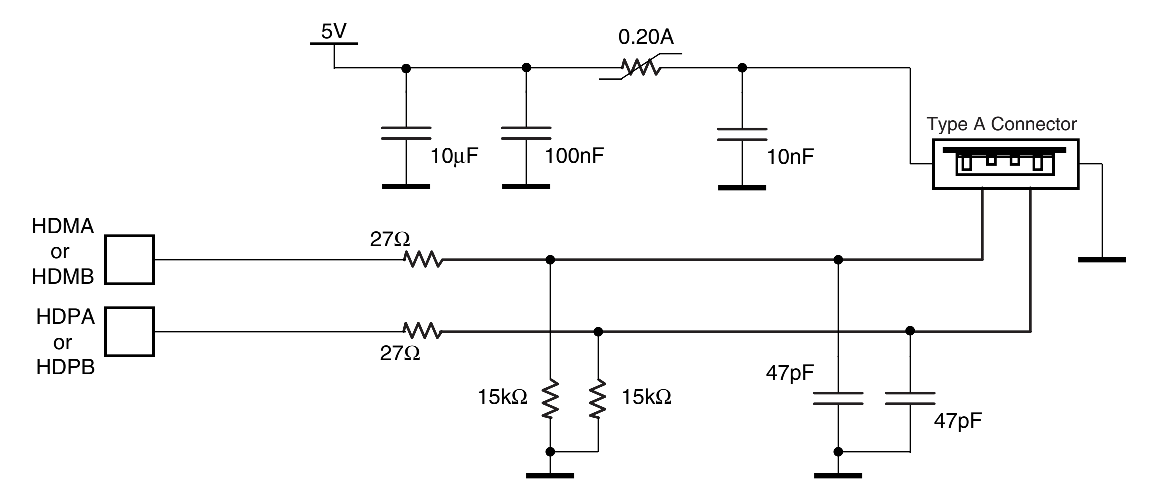


USB Handling is done through several layers as follows:

- Host controller hardware and serial engine: Transmit and receive USB data on the bus.
- Host controller driver: Drives the Host controller hardware and handle the USB protocol
- USB Bus driver and hub driver: Handles USB commands and enumeration. Offers a hardware independent interface.
- Mini driver: Handles device specific commands.
- Class driver: handles standard devices. This acts as a generic driver for a class of devices, for example the HID driver.

35.5 Typical Connection

Figure 35-4. Board Schematic to Interface UHP Device Controller



As device connection is automatically detected by the USB host port logic, a pull-down must be connected on DP and DM on the board. Otherwise the USB host will permanently detect a device connection on this port.

36. USB Device Port (UDP)

36.1 Description

The USB Device Port (UDP) is compliant with the Universal Serial Bus (USB) V2.0 full-speed device specification.

Each endpoint can be configured in one of several USB transfer types. It can be associated with one or two banks of a dual-port RAM used to store the current data payload. If two banks are used, one DPR bank is read or written by the processor, while the other is read or written by the USB device peripheral. This feature is mandatory for isochronous endpoints. Thus the device maintains the maximum bandwidth (1M bytes/s) by working with endpoints with two banks of DPR.

Table 36-1. USB Endpoint Description

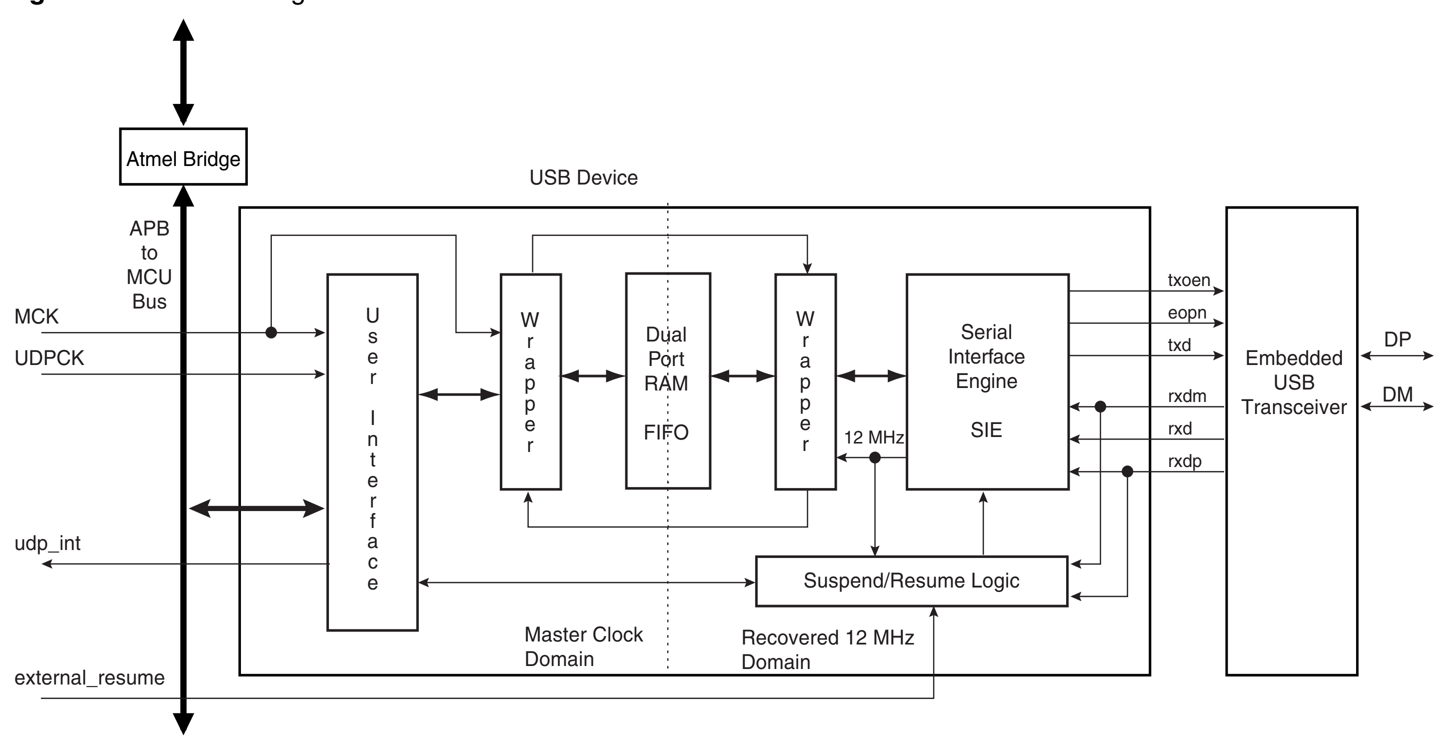
Endpoint Number	Mnemonic	Dual-Bank	Max. Endpoint Size	Endpoint Type
0	EP0	No	8	Control/Bulk/Interrupt
1	EP1	Yes	64	Bulk/Iso/Interrupt
3	EP2	Yes	64	Bulk/Iso/Interrupt
3	EP3	No	64	Control/Bulk/Interrupt
4	EP4	Yes	256	Bulk/Iso/Interrupt
5	EP5	Yes	256	Bulk/Iso/Interrupt

Suspend and resume are automatically detected by the USB device, which notifies the processor by raising an interrupt. Depending on the product, an external signal can be used to send a wake-up to the USB host controller.



36.2 Block Diagram

Figure 36-1. Block Diagram



Access to the UDP is via the APB bus interface. Read and write to the data FIFO are done by reading and writing 8-bit values to APB registers.

The UDP peripheral requires two clocks: one peripheral clock used by the MCK domain and a 48 MHz clock used by the 12 MHz domain.

A USB 2.0 full-speed pad is embedded and controlled by the Serial Interface Engine (SIE).

The signal external_resume is optional. It allows the UDP peripheral to wake-up once in system mode. The host is then notified that the device asks for a resume. This optional feature must be also negotiated with the host during the enumeration.

36.3 Product Dependencies

For further details on the USB Device hardware implementation, see the specific Product Properties document.

The USB physical transceiver is integrated into the product. The bidirectional differential signals DP and DM are available from the product boundary.

Two I/O lines may be used by the application:

- One to check that VBUS is still available from the host. Self-powered devices may use this entry to be notified that the host has been powered off. In this case, the board pull-up on DP must be disabled in order to prevent feeding current to the host.
- One to control the board pull-up on DP. Thus, when the device is ready to communicate with the host, it activates its DP pull-up through this control line.

36.3.1 I/O Lines

DP and DM are not controlled by any PIO controllers. The embedded USB physical transceiver is controlled by the USB device peripheral.

To reserve an I/O line to check VBUS, the programmer must first program the PIO controller to assign this I/O in input PIO mode.

To reserve an I/O line to control the board pull-up, the programmer must first program the PIO controller to assign this I/O in output PIO mode.

36.3.2 Power Management

The USB device peripheral requires a 48 MHz clock. This clock must be generated by a PLL with an accuracy of ± 0.25%.

Thus, the USB device receives two clocks from the Power Management Controller (PMC): the master clock, MCK, used to drive the peripheral user interface, and the UDPCK, used to interface with the bus USB signals (recovered 12 MHz domain).

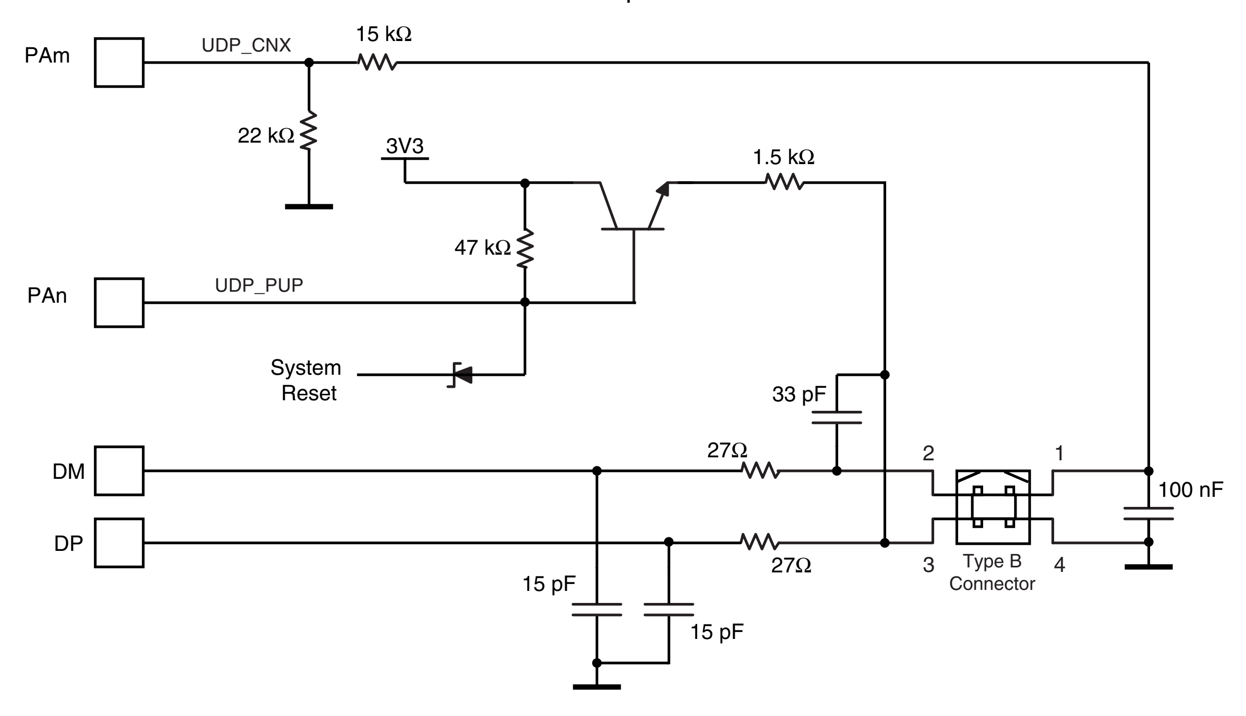
36.3.3 Interrupt

The USB device interface has an interrupt line connected to the Advanced Interrupt Controller (AIC).

Handling the USB device interrupt requires programming the AIC before configuring the UDP.

36.4 Typical Connection

Figure 36-2. Board Schematic to Interface USB Device Peripheral



UDP_ CNX is an input signal used to check if the host is connected

UDP_ PUP is an output signal used to disable pull-up on DP by driving it to 0.

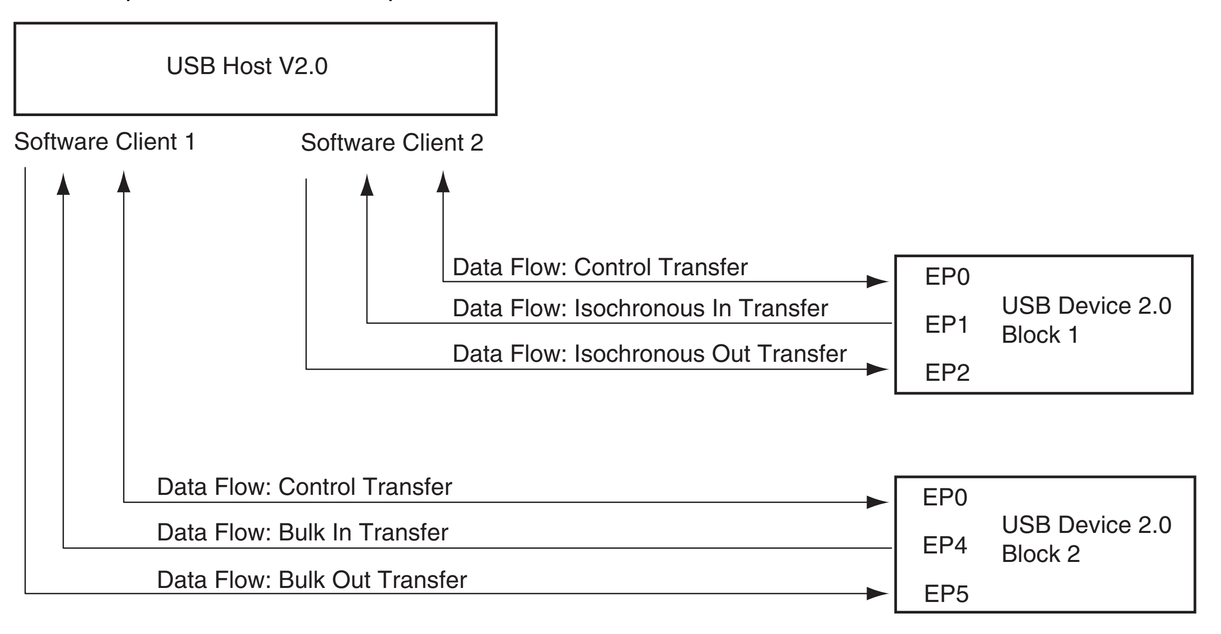
Figure 36-2 shows automatic activation of pull-up after reset.

36.5 Functional Description

36.5.1 USB V2.0 Full-speed Introduction

The USB V2.0 full-speed provides communication services between host and attached USB devices. Each device is offered with a collection of communication flows (pipes) associated with each endpoint. Software on the host communicates with an USB device through a set of communication flows.

Figure 36-3. Example of USB V2.0 Full-speed Communication Control



36.5.1.1 USB V2.0 Full-speed Transfer Types

A communication flow is carried over one of four transfer types defined by the USB device.

Table 36-2. USB Communication Flow

Transfer	Direction	Bandwidth	Endpoint Size	Error Detection	Retrying
Control	Bidirectional	Not guaranteed	8, 16, 32, 64	Yes	Automatic
Isochronous	Unidirectional	Guaranteed	1 - 1023	Yes	No
Interrupt	Unidirectional	Not guaranteed	≤64	Yes	Yes
Bulk	Unidirectional	Not guaranteed	8, 16, 32, 64	Yes	Yes

36.5.1.2 USB Bus Transactions

Each transfer results in one or more transactions over the USB bus. There are five kinds of transactions flowing across the bus in packets:

1. Setup Transaction
2. Data IN Transaction
3. Data OUT Transaction
4. Status IN Transaction
5. Status OUT Transaction

36.5.1.3 USB Transfer Event Definitions

As indicated below, transfers are sequential events carried out on the USB bus.

Table 36-3. USB Transfer Events

Control Transfers[(1) (3)]	• Setup transaction > Data IN transactions > Status OUT transaction • Setup transaction > Data OUT transactions > Status IN transaction • Setup transaction > Status IN transaction
Interrupt IN Transfer (device toward host)	• Data IN transaction > Data IN transaction
Interrupt OUT Transfer (host toward device)	• Data OUT transaction > Data OUT transaction
Isochronous IN Transfer[(2)] (device toward host)	• Data IN transaction > Data IN transaction
Isochronous OUT Transfer[(2)] (host toward device)	• Data OUT transaction > Data OUT transaction
Bulk IN Transfer (device toward host)	• Data IN transaction > Data IN transaction
Bulk OUT Transfer (host toward device)	• Data OUT transaction > Data OUT transaction

Notes: 1. Control transfer must use endpoints with no ping-pong attributes.
2. Isochronous transfers must use endpoints with ping-pong attributes.
3. Control transfers can be aborted using a stall handshake.

36.5.2 Handling Transactions with USB V2.0 Device Peripheral

36.5.2.1 Setup Transaction

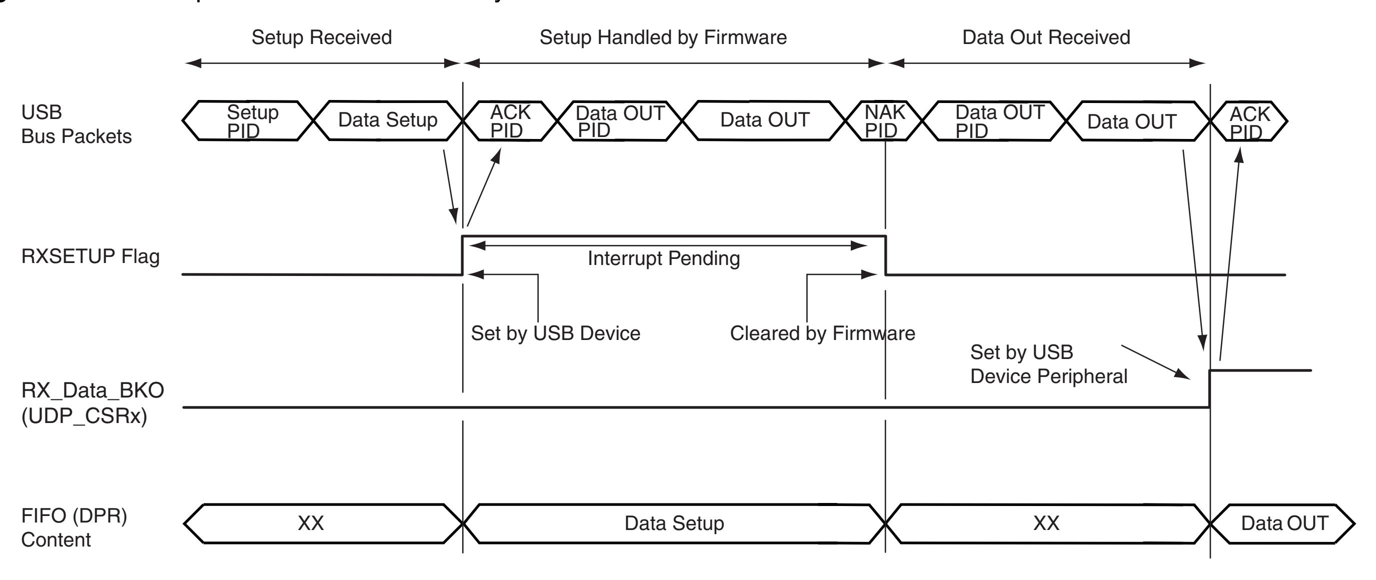
Setup is a special type of host-to-device transaction used during control transfers. Control transfers must be performed using endpoints with no ping-pong attributes. A setup transaction needs to be handled as soon as possible by the firmware. It is used to transmit requests from the host to the device. These requests are then handled by the USB device and may require more arguments. The arguments are sent to the device by a Data OUT transaction which follows the setup transaction. These requests may also return data. The data is carried out to the host by the next Data IN transaction which follows the setup transaction. A status transaction ends the control transfer.

When a setup transfer is received by the USB endpoint:

- The USB device automatically acknowledges the setup packet
- RXSETUP is set in the UDP_ CSRx register
- An endpoint interrupt is generated while the RXSETUP is not cleared. This interrupt is carried out to the microcontroller if interrupts are enabled for this endpoint.

Thus, firmware must detect the RXSETUP polling the UDP_ CSRx or catching an interrupt, read the setup packet in the FIFO, then clear the RXSETUP. RXSETUP cannot be cleared before the setup packet has been read in the FIFO. Otherwise, the USB device would accept the next Data OUT transfer and overwrite the setup packet in the FIFO.

Figure 36-4. Setup Transaction Followed by a Data OUT Transaction



36.5.2.2 Data IN Transaction

Data IN transactions are used in control, isochronous, bulk and interrupt transfers and conduct the transfer of data from the device to the host. Data IN transactions in isochronous transfer must be done using endpoints with ping-pong attributes.

• Using Endpoints Without Ping-pong Attributes

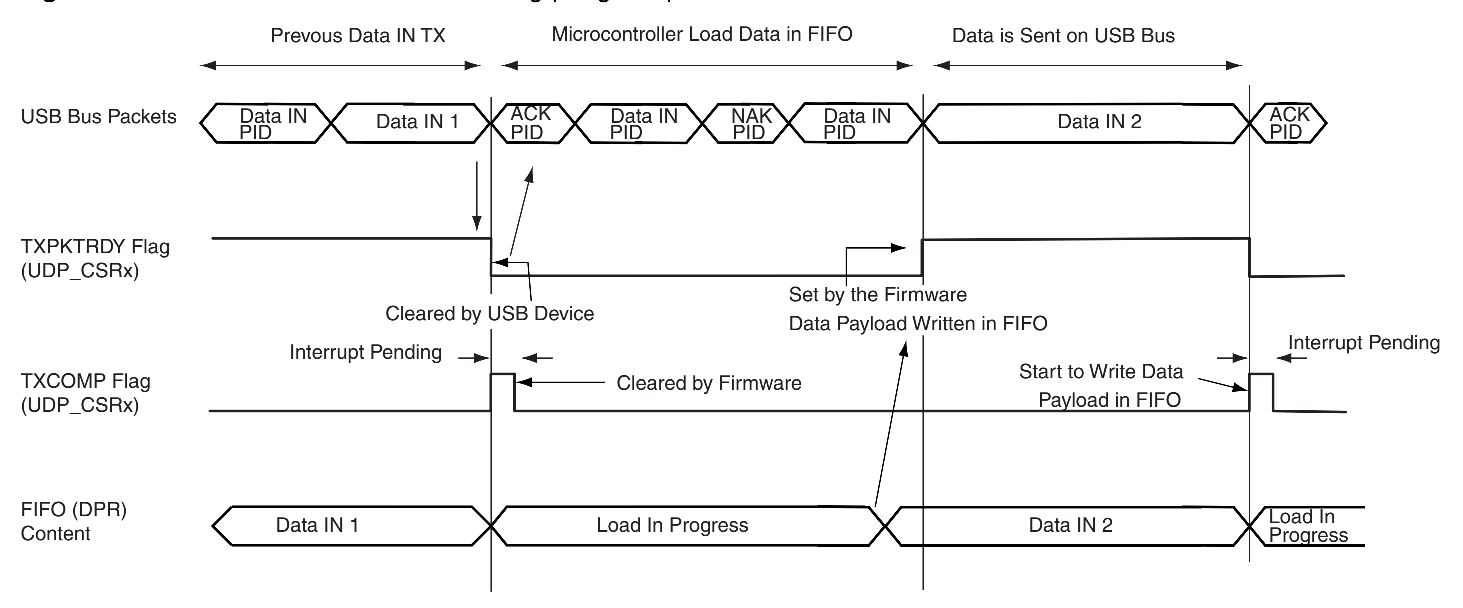
To perform a Data IN transaction using a non ping-pong endpoint:

1. The microcontroller checks if it is possible to write in the FIFO by polling TXPKTRDY in the endpoint's UDP_ CSRx register (TXPKTRDY must be cleared).
2. The microcontroller writes data to be sent in the endpoint's FIFO, writing zero or more byte values in the endpoint's UDP_ FDRx register,
3. The microcontroller notifies the USB peripheral it has finished by setting the TXPK-TRDY in the endpoint's UDP_ CSRx register.
4. The microcontroller is notified that the endpoint's FIFO has been released by the USB device when TXCOMP in the endpoint's UDP_ CSRx register has been set. Then an interrupt for the corresponding endpoint is pending while TXCOMP is set.

TXCOMP is set by the USB device when it has received an ACK PID signal for the Data IN packet. An interrupt is pending while TXCOMP is set.

Note: Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev 2.0,* for more information on the Data IN protocol layer.

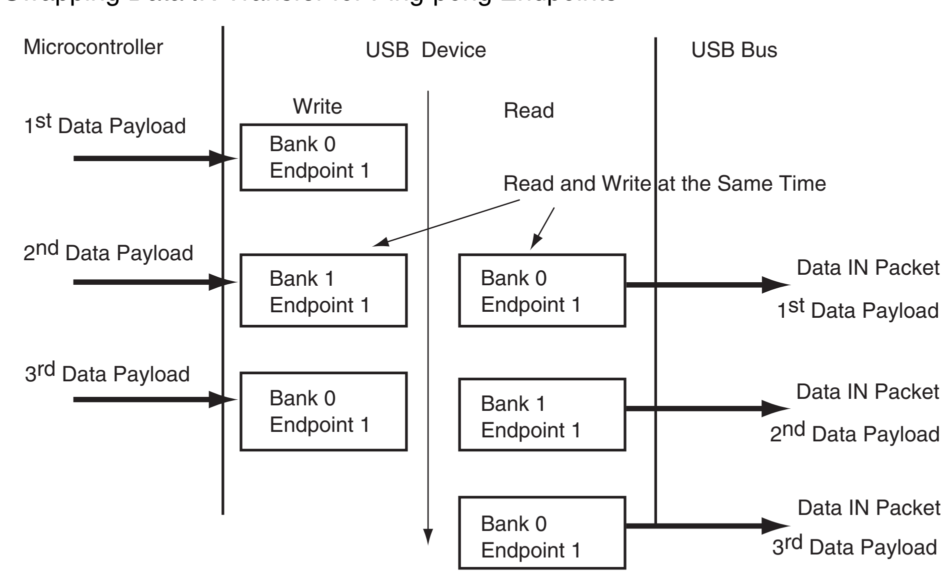
Figure 36-5. Data IN Transfer for Non Ping-pong Endpoint



- Using Endpoints With Ping-pong Attribute

The use of an endpoint with ping-pong attributes is necessary during isochronous transfer. To be able to guarantee a constant bandwidth, the microcontroller must prepare the next data payload to be sent while the current one is being sent by the USB device. Thus two banks of memory are used. While one is available for the microcontroller, the other one is locked by the USB device.

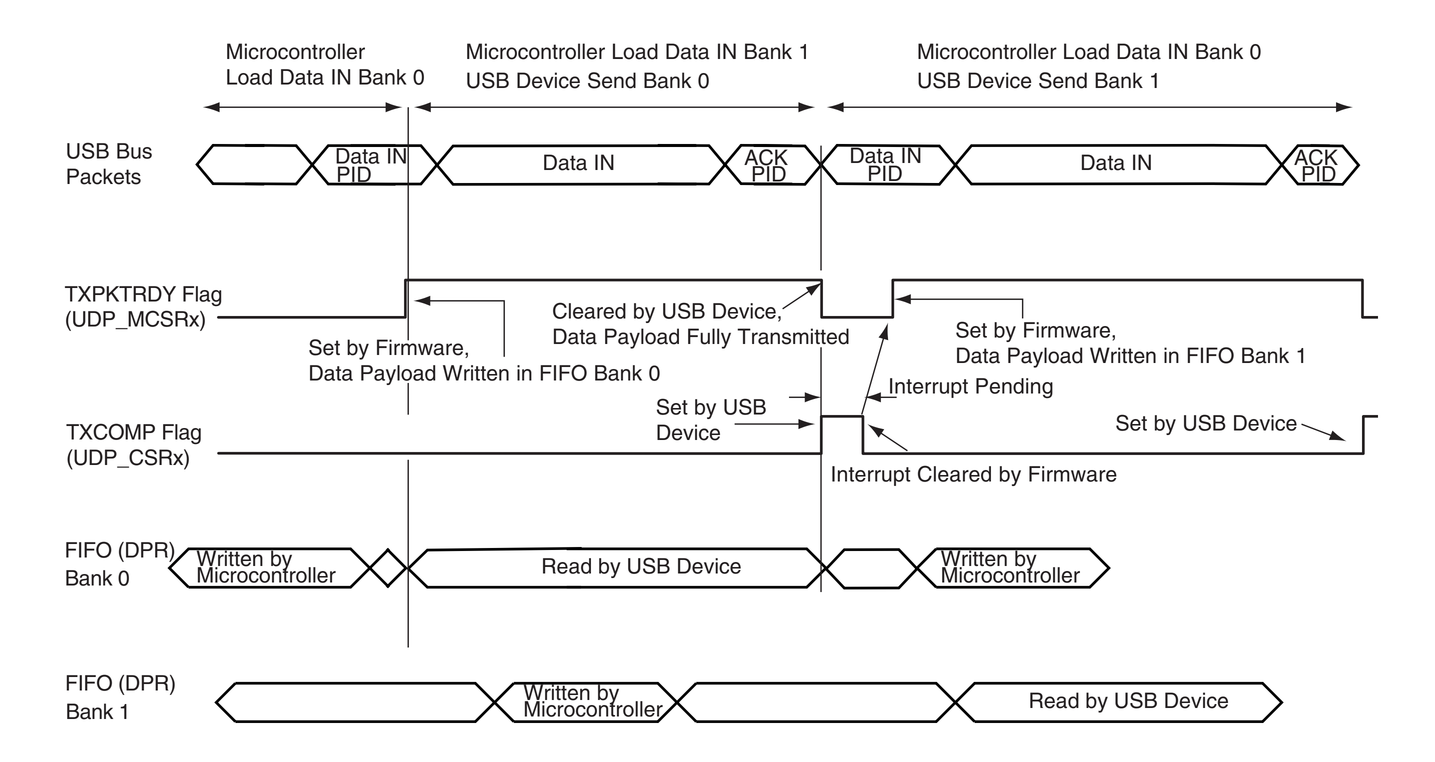
Figure 36-6. Bank Swapping Data IN Transfer for Ping-pong Endpoints



When using a ping-pong endpoint, the following procedures are required to perform Data IN transactions:

1. The microcontroller checks if it is possible to write in the FIFO by polling TXPKTRDY to be cleared in the endpoint's UDP_ CSRx register.
2. The microcontroller writes the first data payload to be sent in the FIFO (Bank 0), writing zero or more byte values in the endpoint's UDP_ FDRx register.
3. The microcontroller notifies the USB peripheral it has finished writing in Bank 0 of the FIFO by setting the TXPKTRDY in the endpoint's UDP_ CSRx register.
4. Without waiting for TXPKTRDY to be cleared, the microcontroller writes the second data payload to be sent in the FIFO (Bank 1), writing zero or more byte values in the endpoint's UDP_ FDRx register.
5. The microcontroller is notified that the first Bank has been released by the USB device when TXCOMP in the endpoint's UDP_ CSRx register is set. An interrupt is pending while TXCOMP is being set.
6. Once the microcontroller has received TXCOMP for the first Bank, it notifies the USB device that it has prepared the second Bank to be sent rising TXPKTRDY in the endpoint's UDP_ CSRx register.
7. At this step, Bank 0 is available and the microcontroller can prepare a third data payload to be sent.

Figure 36-7. Data IN Transfer for Ping-pong Endpoint



Warning: There is software critical path due to the fact that once the second bank is filled, the driver has to wait for TX_COMP to set TX_PKTRDY. If the delay between receiving TX_COMP is set and TX_PKTRDY is set is too long, some Data IN packets may be NACKed, reducing the bandwidth.

36.5.2.3 Data OUT Transaction

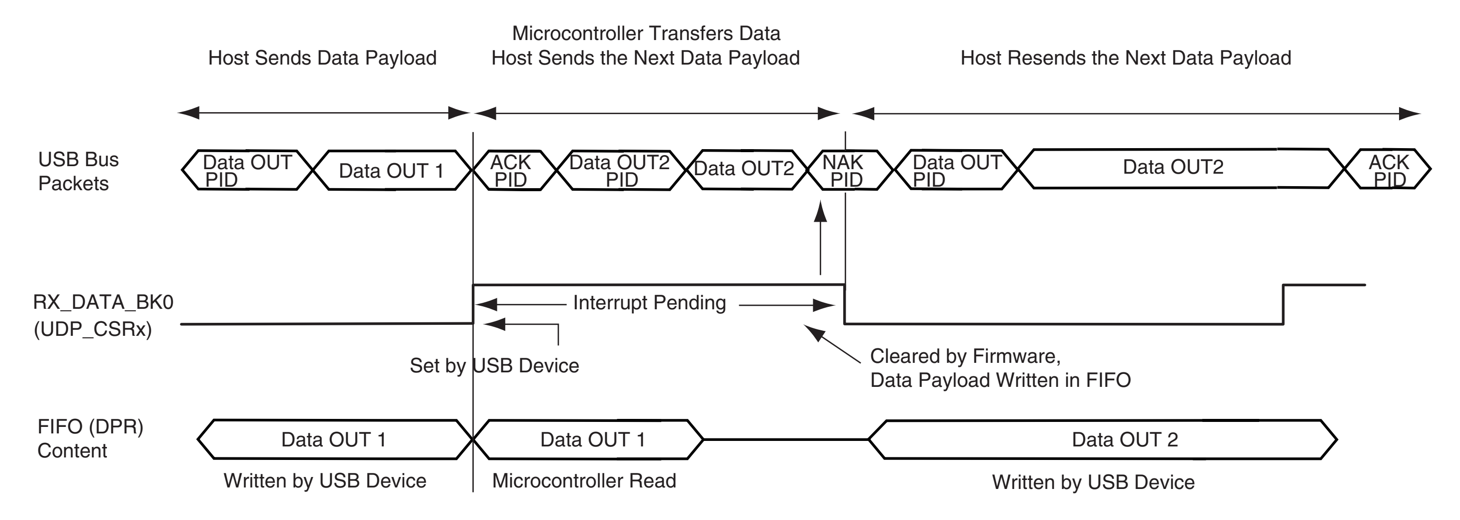
Data OUT transactions are used in control, isochronous, bulk and interrupt transfers and conduct the transfer of data from the host to the device. Data OUT transactions in isochronous transfers must be done using endpoints with ping-pong attributes.

• Data OUT Transaction Without Ping-pong Attributes

To perform a Data OUT transaction, using a non ping-pong endpoint:

1. The host generates a Data OUT packet.
2. This packet is received by the USB device endpoint. While the FIFO associated to this endpoint is being used by the microcontroller, a NAK PID is returned to the host. Once the FIFO is available, data are written to the FIFO by the USB device and an ACK is automatically carried out to the host.
3. The microcontroller is notified that the USB device has received a data payload polling RX_DATA_BK0 in the endpoint's UDP_ CSRx register. An interrupt is pending for this endpoint while RX_DATA_BK0 is set.
4. The number of bytes available in the FIFO is made available by reading RXBYTECNT in the endpoint's UDP_ CSRx register.
5. The microcontroller carries out data received from the endpoint's memory to its memory. Data received is available by reading the endpoint's UDP_ FDRx register.
6. The microcontroller notifies the USB device that it has finished the transfer by clearing RX_DATA_BK0 in the endpoint's UDP_ CSRx register.
7. A new Data OUT packet can be accepted by the USB device.

Figure 36-8. Data OUT Transfer for Non Ping-pong Endpoints



An interrupt is pending while the flag RX_DATA_BK0 is set. Memory transfer between the USB device, the FIFO and microcontroller memory can not be done after RX_DATA_BK0 has been cleared. Otherwise, the USB device would accept the next Data OUT transfer and overwrite the current Data OUT packet in the FIFO.

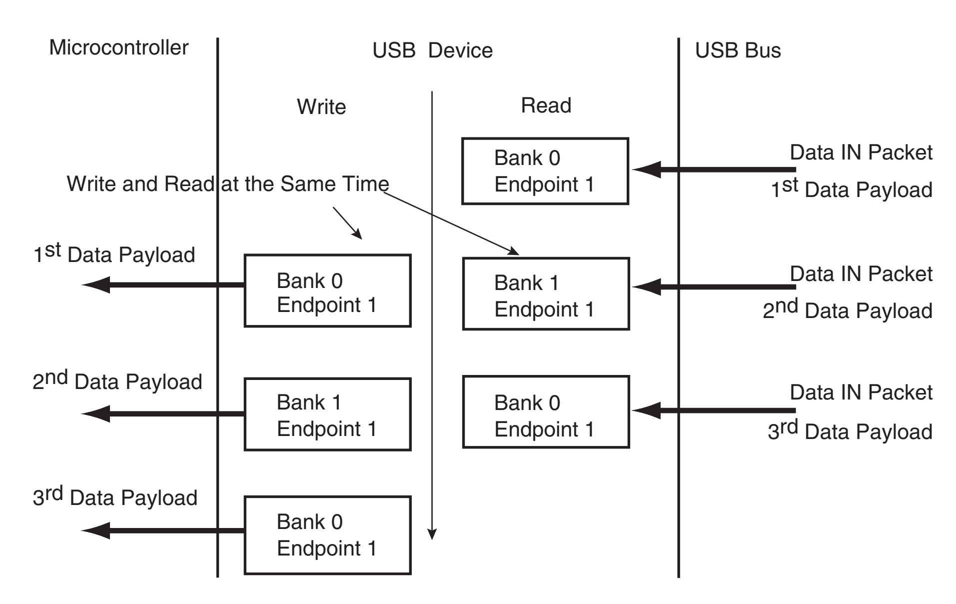
• Using Endpoints With Ping-pong Attributes

During isochronous transfer, using an endpoint with ping-pong attributes is obligatory. To be able to guarantee a constant bandwidth, the microcontroller must read the previous data pay-



load sent by the host, while the current data payload is received by the USB device. Thus two banks of memory are used. While one is available for the microcontroller, the other one is locked by the USB device.

Figure 36-9. Bank Swapping in Data OUT Transfers for Ping-pong Endpoints



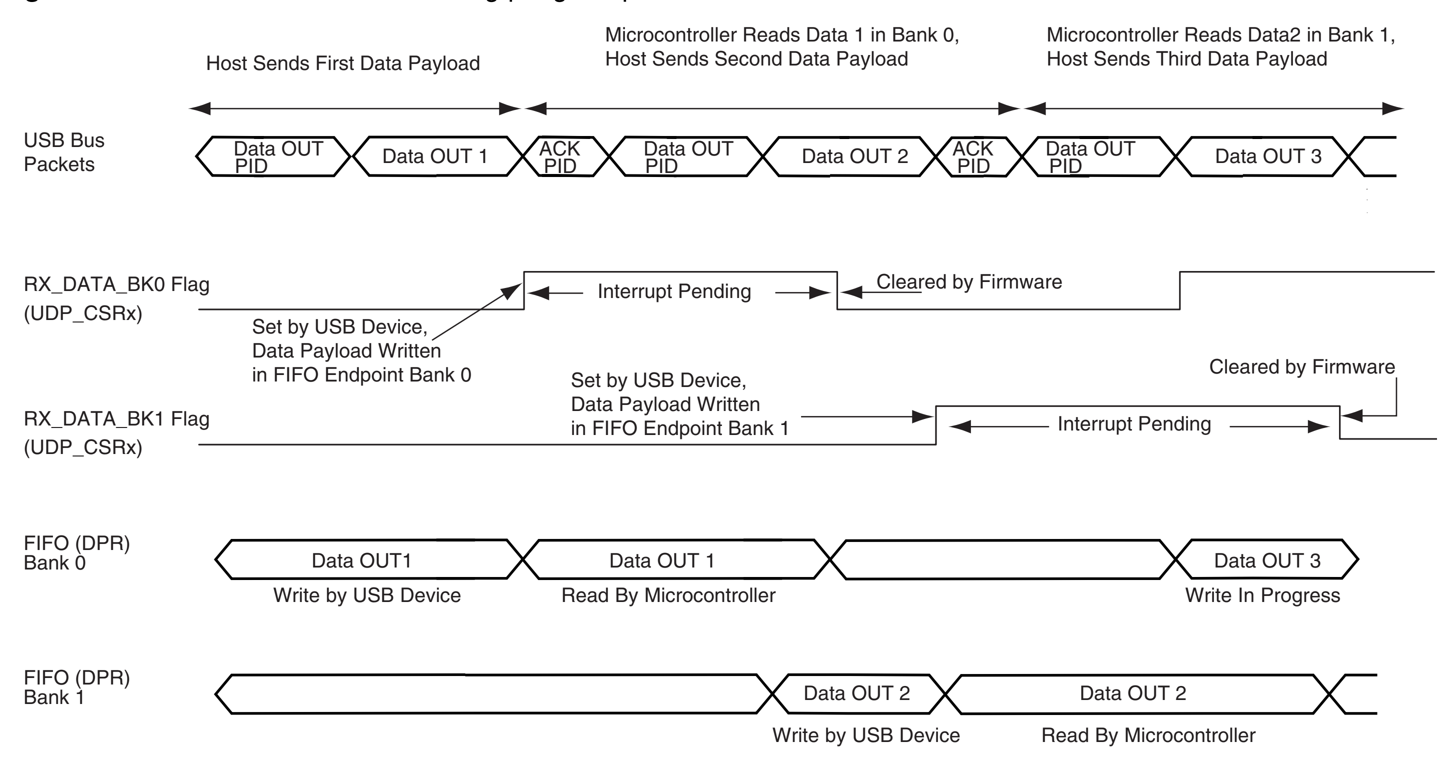
When using a ping-pong endpoint, the following procedures are required to perform Data OUT transactions:

1. The host generates a Data OUT packet.
2. This packet is received by the USB device endpoint. It is written in the endpoint's FIFO Bank 0.
3. The USB device sends an ACK PID packet to the host. The host can immediately send a second Data OUT packet. It is accepted by the device and copied to FIFO Bank 1.
4. The microcontroller is notified that the USB device has received a data payload, polling RX_DATA_BK0 in the endpoint's UDP_ CSRx register. An interrupt is pending for this endpoint while RX_DATA_BK0 is set.
5. The number of bytes available in the FIFO is made available by reading RXBYTECNT in the endpoint's UDP_ CSRx register.
6. The microcontroller transfers out data received from the endpoint's memory to the microcontroller's memory. Data received is made available by reading the endpoint's UDP_ FDRx register.
7. The microcontroller notifies the USB peripheral device that it has finished the transfer by clearing RX_DATA_BK0 in the endpoint's UDP_ CSRx register.
8. A third Data OUT packet can be accepted by the USB peripheral device and copied in the FIFO Bank 0.
9. If a second Data OUT packet has been received, the microcontroller is notified by the flag RX_DATA_BK1 set in the endpoint's UDP_ CSRx register. An interrupt is pending for this endpoint while RX_DATA_BK1 is set.
10. The microcontroller transfers out data received from the endpoint's memory to the microcontroller's memory. Data received is available by reading the endpoint's UDP_ FDRx register.



11. The microcontroller notifies the USB device it has finished the transfer by clearing RX_DATA_BK1 in the endpoint's UDP_ CSRx register.
12. A fourth Data OUT packet can be accepted by the USB device and copied in the FIFO Bank 0.

Figure 36-10. Data OUT Transfer for Ping-pong Endpoint



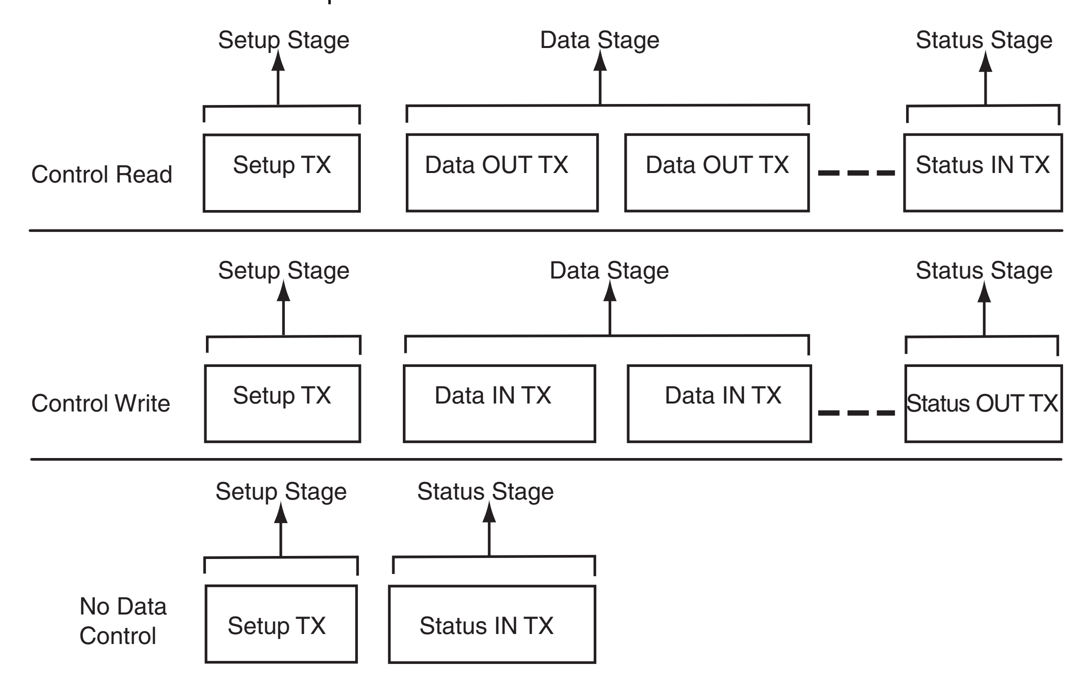
Note: An interrupt is pending while the RX_DATA_BK0 or RX_DATA_BK1 flag is set.

Warning: When RX_DATA_BK0 and RX_DATA_BK1 are both set, there is no way to determine which one to clear first. Thus the software must keep an internal counter to be sure to clear alternatively RX_DATA_BK0 then RX_DATA_BK1. This situation may occur when the software application is busy elsewhere and the two banks are filled by the USB host. Once the application comes back to the USB driver, the two flags are set.

36.5.2.4 Status Transaction

A status transaction is a special type of host-to-device transaction used only in a control transfer. The control transfer must be performed using endpoints with no ping-pong attributes. According to the control sequence (read or write), the USB device sends or receives a status transaction.

Figure 36-11. Control Read and Write Sequences



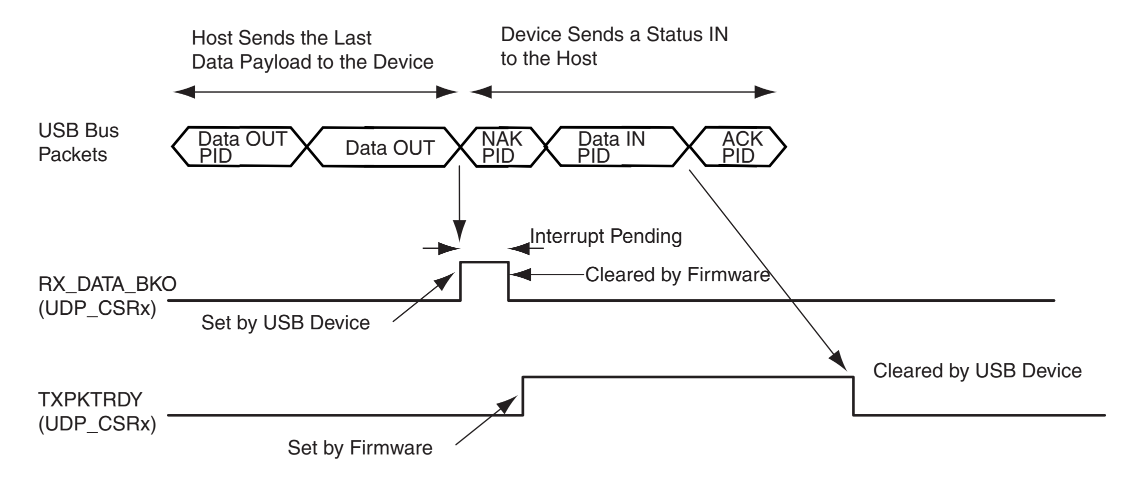
Notes: 1. During the Status IN stage, the host waits for a zero length packet (Data IN transaction with no data) from the device using DATA1 PID. Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev. 2.0,* for more information on the protocol layer.
2. During the Status OUT stage, the host emits a zero length packet to the device (Data OUT transaction with no data).

36.5.2.5 Status IN Transfer

Once a control request has been processed, the device returns a status to the host. This is a zero length Data IN transaction.

1. The microcontroller waits for TXPKTRDY in the UDP_ CSRx endpoint's register to be cleared. (At this step, TXPKTRDY must be cleared because the previous transaction was a setup transaction or a Data OUT transaction.)
2. Without writing anything to the UDP_ FDRx endpoint's register, the microcontroller sets TXPKTRDY. The USB device generates a Data IN packet using DATA1 PID.
3. This packet is acknowledged by the host and TXPKTRDY is set in the UDP_ CSRx endpoint's register.

Figure 36-12. Data Out Followed by Status IN Transfer.

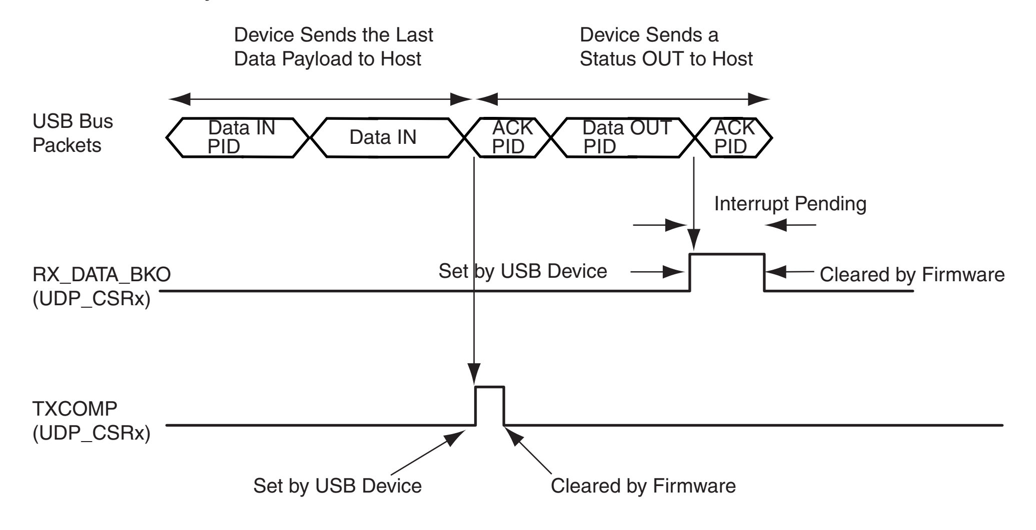


36.5.2.6 *Status OUT Transfer*

Once a control request has been processed and the requested data returned, the host acknowledges by sending a zero length packet. This is a zero length Data OUT transaction.

1. The USB device receives a zero length packet. It sets RX_DATA_BK0 flag in the UDP_ CSRx register and acknowledges the zero length packet.
2. The microcontroller is notified that the USB device has received a zero length packet sent by the host polling RX_DATA_BK0 in the UDP_ CSRx register. An interrupt is pending while RX_DATA_BK0 is set. The number of bytes received in the endpoint's UDP_ BCR register is equal to zero.
3. The microcontroller must clear RX_DATA_BK0.

Figure 36-13. Data IN Followed by Status OUT Transfer



36.5.2.7 *Stall Handshake*

A stall handshake can be used in one of two distinct occasions. (For more information on the stall handshake, refer to Chapter 8 of the *Universal Serial Bus Specification, Rev 2.0.*)



- A functional stall is used when the halt feature associated with the endpoint is set. (Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev 2.0,* for more information on the halt feature.)
- To abort the current request, a protocol stall is used, but uniquely with control transfer.

The following procedure generates a stall packet:

1. The microcontroller sets the FORCESTALL flag in the UDP_ CSRx endpoint's register.
2. The host receives the stall packet.
3. The microcontroller is notified that the device has sent the stall by polling the STALLSENT to be set. An endpoint interrupt is pending while STALLSENT is set. The microcontroller must clear STALLSENT to clear the interrupt.

When a setup transaction is received after a stall handshake, STALLSENT must be cleared in order to prevent interrupts due to STALLSENT being set.

Figure 36-14. Stall Handshake (Data IN Transfer)

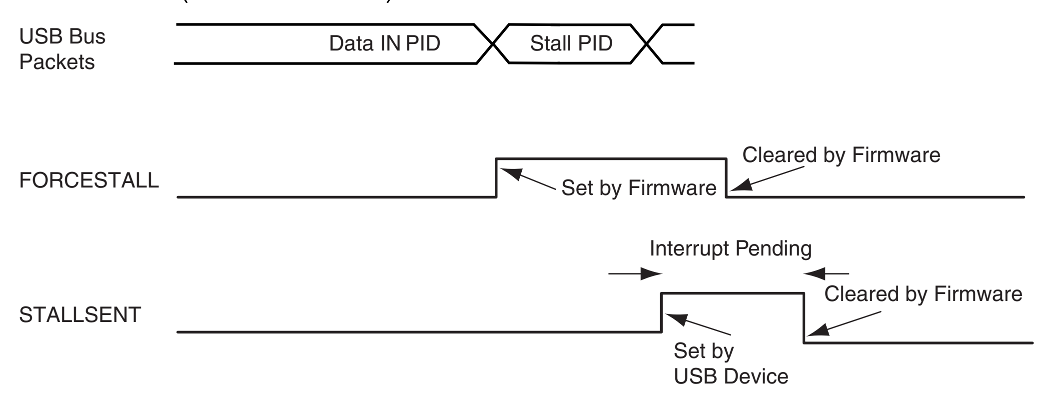
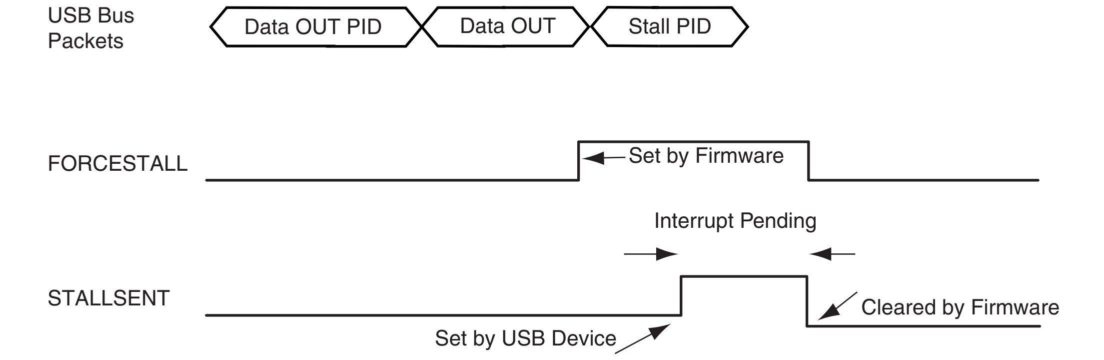


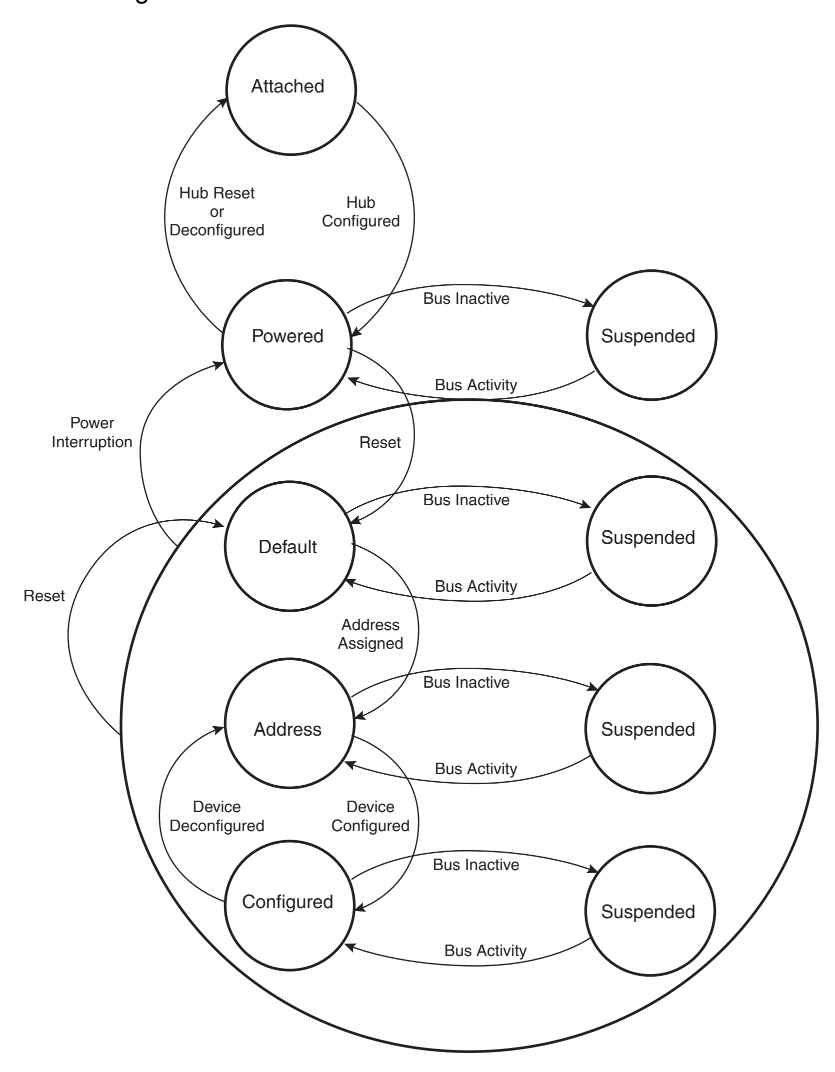
Figure 36-15. Stall Handshake (Data OUT Transfer)



36.5.3 Controlling Device States

A USB device has several possible states. Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev 2.0*.

Figure 36-16. USB Device State Diagram



Movement from one state to another depends on the USB bus state or on standard requests sent through control transactions via the default endpoint (endpoint 0).

After a period of bus inactivity, the USB device enters Suspend Mode. Accepting Suspend/Resume requests from the USB host is mandatory. Constraints in Suspend Mode are very strict for bus-powered applications; devices may not consume more than 500 uA on the USB bus.

While in Suspend Mode, the host may wake up a device by sending a resume signal (bus activity) or a USB device may send a wake-up request to the host, e.g., waking up a PC by moving a USB mouse.

The wake-up feature is not mandatory for all devices and must be negotiated with the host.

36.5.3.1 From Powered State to Default State

After its connection to a USB host, the USB device waits for an end-of-bus reset. The USB host stops driving a reset state once it has detected the device's pull-up on DP. The unmasked flag ENDBURSES is set in the register UDP_ISR and an interrupt is triggered. The UDP software enables the default endpoint, setting the EPEDS flag in the UDP_CSR[0] register and, optionally, enabling the interrupt for endpoint 0 by writing 1 to the UDP_IER register. The enumeration then begins by a control transfer.

36.5.3.2 From Default State to Address State

After a set address standard device request, the USB host peripheral enters the address state. Before this, it achieves the Status IN transaction of the control transfer, i.e., the UDP device sets its new address once the TXCOMP flag in the UDP_CSR[0] register has been received and cleared.

To move to address state, the driver software sets the FADDEN flag in the UDP_GLB_STATE, sets its new address, and sets the FEN bit in the UDP_FADDR register.

36.5.3.3 From Address State to Configured State

Once a valid Set Configuration standard request has been received and acknowledged, the device enables endpoints corresponding to the current configuration. This is done by setting the EPEDS and EPTYPE fields in the UDP_CSRx registers and, optionally, enabling corresponding interrupts in the UDP_IER register.

36.5.3.4 Enabling Suspend

When a Suspend (no bus activity on the USB bus) is detected, the RXSUSP signal in the UDP_ISR register is set. This triggers an interrupt if the corresponding bit is set in the UDP_IMR register.

This flag is cleared by writing to the UDP_ICR register. Then the device enters Suspend Mode. As an example, the microcontroller switches to slow clock, disables the PLL and main oscillator, and goes into Idle Mode. It may also switch off other devices on the board.

The USB device peripheral clocks may be switched off. However, the transceiver and the USB peripheral must not be switched off, otherwise the resume is not detected.

36.5.3.5 Receiving a Host Resume

In suspend mode, the USB transceiver and the USB peripheral must be powered to detect the RESUME. However, the USB device peripheral may not be clocked as the WAKEUP signal is asynchronous.

Once the resume is detected on the bus, the signal WAKEUP in the UDP_ISR is set. It may generate an interrupt if the corresponding bit in the UDP_IMR register is set. This interrupt may be used to wake-up the core, enable PLL and main oscillators and configure clocks. The WAKEUP bit must be cleared as soon as possible by setting WAKEUP in the UDP_ICR register.

36.5.3.6 Sending an External Resume

The External Resume is negotiated with the host and enabled by setting the ESR bit in the UDP_ GLB_STATE. An asynchronous event on the ext_resume_pin of the peripheral generates a WAKEUP interrupt. On early versions of the USP peripheral, the K-state on the USB line is generated immediately. This means that the USB device must be able to answer to the host very quickly. On recent versions, the software sets the RMWUPE bit in the

UDP_GLB_STATE register once it is ready to communicate with the host. The K-state on the bus is then generated.

The WAKEUP bit must be cleared as soon as possible by setting WAKEUP in the UDP_ICR register.

36.6 USB Device Port (UDP) User Interface

Table 36-4. UDP Memory Map

Offset	Register	Name	Access	Reset State
0x000	Frame Number Register	UDP_ FRM_NUM	Read	0x0000_0000
0x004	Global State Register	UDP_ GLB_STAT	Read/Write	0x0000_0010
0x008	Function Address Register	UDP_ FADDR	Read/Write	0x0000_0100
0x00C	Reserved	–	–	–
0x010	Interrupt Enable Register	UDP_ IER	Write	
0x014	Interrupt Disable Register	UDP_ IDR	Write	
0x018	Interrupt Mask Register	UDP_ IMR	Read	0x0000_1200
0x01C	Interrupt Status Register	UDP_ ISR	Read	0x0000_XX00
0x020	Interrupt Clear Register	UDP_ ICR	Write	
0x024	Reserved	–	–	–
0x028	Reset Endpoint Register	UDP_ RST_EP	Read/Write	
0x02C	Reserved	–	–	–
0x030	Endpoint 0 Control and Status Register	UDP_CSR0	Read/Write	0x0000_0000
. . .	. . .			
See Note: [1]	Endpoint 5 Control and Status Register	UDP_CSR5	Read/Write	0x0000_0000
0x050	Endpoint 0 FIFO Data Register	UDP_ FDR0	Read/Write	0x0000_0000
. . .	. . .			
See Note: [2]	Endpoint 5 FIFO Data Register	UDP_ FDR5	Read/Write	0x0000_0000
0x070	Reserved	–	–	–
0x074	Transceiver Control Register	UDP_ TXVC	Read/Write	0x0000_0000
0x078 - 0xFC	Reserved	–	–	–

Notes: 1. The addresses of the UDP_ CSRx registers are calculated as: 0x030 + 4(Endpoint Number - 1).
2. The addresses of the UDP_ FDRx registers are calculated as: 0x050 + 4(Endpoint Number - 1).

36.6.1 UDP Frame Number Register

Register Name: UDP_ FRM_NUM

Access Type: Read-only

31	30	29	28	27	26	25	24
---	---	---	---	---	---	---	---

23	22	21	20	19	18	17	16
–	–	–	–	–	–	FRM_OK	FRM_ERR

15	14	13	12	11	10	9	8
–	–	–	–	–	FRM_NUM		

7	6	5	4	3	2	1	0
FRM_NUM							

- **FRM_NUM[10:0]: Frame Number as Defined in the Packet Field Formats**

This 11-bit value is incremented by the host on a per frame basis. This value is updated at each start of frame.

Value Updated at the SOF_EOP (Start of Frame End of Packet).

- **FRM_ERR: Frame Error**

This bit is set at SOF_EOP when the SOF packet is received containing an error.

This bit is reset upon receipt of SOF_PID.

- **FRM_OK: Frame OK**

This bit is set at SOF_EOP when the SOF packet is received without any error.

This bit is reset upon receipt of SOF_PID (Packet Identification).

In the Interrupt Status Register, the SOF interrupt is updated upon receiving SOF_PID. This bit is set without waiting for EOP.

Note: In the 8-bit Register Interface, FRM_OK is bit 4 of FRM_NUM_H and FRM_ERR is bit 3 of FRM_NUM_L.

36.6.2 UDP Global State Register

Register Name: UDP_ GLB_STAT

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	RMWUPE	RSMINPR	ESR	CONFG	FADDEN

This register is used to get and set the device state as specified in Chapter 9 of the *USB Serial Bus Specification, Rev.2.0*.

- **FADDEN: Function Address Enable**

Read:

0 = Device is not in address state.

1 = Device is in address state.

Write:

0 = No effect, only a reset can bring back a device to the default state.

1 = Sets device in address state. This occurs after a successful Set Address request. Beforehand, the UDP_ FADDR register must have been initialized with Set Address parameters. Set Address must complete the Status Stage before setting FADDEN. Refer to chapter 9 of the *Universal Serial Bus Specification, Rev. 2.0* for more details.

- **CONFG: Configured**

Read:

0 = Device is not in configured state.

1 = Device is in configured state.

Write:

0 = Sets device in a non configured state

1 = Sets device in configured state.

The device is set in configured state when it is in address state and receives a successful Set Configuration request. Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev. 2.0* for more details.

- **ESR: Enable Send Resume**

0 = Disables the Remote Wake Up sequence.

1 = Remote Wake Up can be processed and the pin send_resume is enabled.

- **RSMINPR: A Resume Has Been Sent to the Host**

Read:

0 = No effect.

1 = A Resume has been received from the host during Remote Wake Up feature.

- **RMWUPE: Remote Wake Up Enable**

0 = Must be cleared after receiving any HOST packet or SOF interrupt.

1 = Enables the K-state on the USB cable if ESR is enabled.

36.6.3 UDP Function Address Register

Register Name: UDP_ FADDR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	FEN

7	6	5	4	3	2	1	0
–	FADD						

- **FADD[6:0]: Function Address Value**

The Function Address Value must be programmed by firmware once the device receives a set address request from the host, and has achieved the status stage of the no-data control sequence. Refer to the *Universal Serial Bus Specification, Rev. 2.0* for more information. After power up or reset, the function address value is set to 0.

- **FEN: Function Enable**

Read:

0 = Function endpoint disabled.

1 = Function endpoint enabled.

Write:

0 = Disables function endpoint.

1 = Default value.

The Function Enable bit (FEN) allows the microcontroller to enable or disable the function endpoints. The microcontroller sets this bit after receipt of a reset from the host. Once this bit is set, the USB device is able to accept and transfer data packets from and to the host.

36.6.4 UDP Interrupt Enable Register

Register Name: UDP_ IER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	WAKEUP	–	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
–	–	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: Enable Endpoint 0 Interrupt**
- **EP1INT: Enable Endpoint 1 Interrupt**
- **EP2INT: Enable Endpoint 2Interrupt**
- **EP3INT: Enable Endpoint 3 Interrupt**
- **EP4INT: Enable Endpoint 4 Interrupt**
- **EP5INT: Enable Endpoint 5 Interrupt**

0 = No effect.

1 = Enables corresponding Endpoint Interrupt.

- **RXSUSP: Enable UDP Suspend Interrupt**

0 = No effect.

1 = Enables UDP Suspend Interrupt.

- **RXRSM: Enable UDP Resume Interrupt**

0 = No effect.

1 = Enables UDP Resume Interrupt.

- **EXTRSM: Enable External Resume Interrupt**

0 = No effect.

1 = Enables External Resume Interrupt.

- **SOFINT: Enable Start Of Frame Interrupt**

0 = No effect.

1 = Enables Start Of Frame Interrupt.

- **WAKEUP: Enable UDP bus Wakeup Interrupt**

0 = No effect.

1 = Enables USB bus Interrupt.

36.6.5 UDP Interrupt Disable Register

Register Name: UDP_ IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	WAKEUP	–	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
–	–	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: Disable Endpoint 0 Interrupt**
- **EP1INT: Disable Endpoint 1 Interrupt**
- **EP2INT: Disable Endpoint 2 Interrupt**
- **EP3INT: Disable Endpoint 3 Interrupt**
- **EP4INT: Disable Endpoint 4 Interrupt**
- **EP5INT: Disable Endpoint 5 Interrupt**

0 = No effect.

1 = Disables corresponding Endpoint Interrupt.

- **RXSUSP: Disable UDP Suspend Interrupt**

0 = No effect.

1 = Disables UDP Suspend Interrupt.

- **RXRSM: Disable UDP Resume Interrupt**

0 = No effect.

1 = Disables UDP Resume Interrupt.

- **EXTRSM: Disable External Resume Interrupt**

0 = No effect.

1 = Disables External Resume Interrupt.

- **SOFINT: Disable Start Of Frame Interrupt**

0 = No effect.

1 = Disables Start Of Frame Interrupt

- **WAKEUP: Disable USB Bus Interrupt**

0 = No effect.

1 = Disables USB Bus Wakeup Interrupt.

36.6.6 UDP Interrupt Mask Register

Register Name: UDP_ IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	WAKEUP	–	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
–	–	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: Mask Endpoint 0 Interrupt**
- **EP1INT: Mask Endpoint 1 Interrupt**
- **EP2INT: Mask Endpoint 2 Interrupt**
- **EP3INT: Mask Endpoint 3 Interrupt**
- **EP4INT: Mask Endpoint 4 Interrupt**
- **EP5INT: Mask Endpoint 5 Interrupt**

0 = Corresponding Endpoint Interrupt is disabled.

1 = Corresponding Endpoint Interrupt is enabled.

- **RXSUSP: Mask UDP Suspend Interrupt**

0 = UDP Suspend Interrupt is disabled.

1 = UDP Suspend Interrupt is enabled.

- **RXRSM: Mask UDP Resume Interrupt.**

0 = UDP Resume Interrupt is disabled.

1 = UDP Resume Interrupt is enabled.

- **EXTRSM: Mask External Resume Interrupt**

0 = External Resume Interrupt is disabled.

1 = External Resume Interrupt is enabled.

- **SOFINT: Mask Start Of Frame Interrupt**

0 = Start of Frame Interrupt is disabled.

1 = Start of Frame Interrupt is enabled.

- **WAKEUP: USB Bus WAKEUP Interrupt**

0 = USB Bus Wakeup Interrupt is disabled.

1 = USB Bus Wakeup Interrupt is enabled.

Note: When the USB block is in suspend mode, the application may power down the USB logic. In this case, any USB HOST resume request that is made must be taken into account and, thus, the reset value of the RXRSM bit of the register UDP_ IMR is enabled.

36.6.7 UDP Interrupt Status Register

Register Name: UDP_ ISR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	WAKEUP	ENDBUSRES	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
–	–	EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: Endpoint 0 Interrupt Status**

0 = No Endpoint0 Interrupt pending.

1 = Endpoint0 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading UDP_ CSR0:

RXSETUP set to 1

RX_DATA_BK0 set to 1

RX_DATA_BK1 set to 1

TXCOMP set to 1

STALLSENT set to 1

EP0INT is a sticky bit. Interrupt remains valid until EP0INT is cleared by writing in the corresponding UDP_ CSR0 bit.

- **EP1INT: Endpoint 1 Interrupt Status**

0 = No Endpoint1 Interrupt pending.

1 = Endpoint1 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading UDP_ CSR1:

RXSETUP set to 1

RX_DATA_BK0 set to 1

RX_DATA_BK1 set to 1

TXCOMP set to 1

STALLSENT set to 1

EP1INT is a sticky bit. Interrupt remains valid until EP1INT is cleared by writing in the corresponding UDP_ CSR1 bit.

- **EP2INT: Endpoint 2 Interrupt Status**

0 = No Endpoint2 Interrupt pending.

1 = Endpoint2 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading UDP_ CSR2:

RXSETUP set to 1

RX_DATA_BK0 set to 1

RX_DATA_BK1 set to 1

TXCOMP set to 1

STALLSENT set to 1

EP2INT is a sticky bit. Interrupt remains valid until EP2INT is cleared by writing in the corresponding UDP_ CSR2 bit.

- **EP3INT: Endpoint 3 Interrupt Status**

0 = No Endpoint3 Interrupt pending.

1 = Endpoint3 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading UDP_ CSR3:

RXSETUP set to 1

RX_DATA_BK0 set to 1

RX_DATA_BK1 set to 1

TXCOMP set to 1

STALLSENT set to 1

EP3INT is a sticky bit. Interrupt remains valid until EP3INT is cleared by writing in the corresponding UDP_ CSR3 bit.

- **EP4INT: Endpoint 4 Interrupt Status**

0 = No Endpoint4 Interrupt pending.

1 = Endpoint4 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading UDP_ CSR4:

RXSETUP set to 1

RX_DATA_BK0 set to 1

RX_DATA_BK1 set to 1

TXCOMP set to 1

STALLSENT set to 1

EP4INT is a sticky bit. Interrupt remains valid until EP4INT is cleared by writing in the corresponding UDP_ CSR4 bit.

- **EP5INT: Endpoint 5 Interrupt Status**

0 = No Endpoint5 Interrupt pending.

1 = Endpoint5 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading UDP_ CSR5:

RXSETUP set to 1

RX_DATA_BK0 set to 1

RX_DATA_BK1 set to 1

TXCOMP set to 1

STALLSENT set to 1

EP5INT is a sticky bit. Interrupt remains valid until EP5INT is cleared by writing in the corresponding UDP_ CSR5 bit.

- **RXSUSP: UDP Suspend Interrupt Status**

0 = No UDP Suspend Interrupt pending.

1 = UDP Suspend Interrupt has been raised.

The USB device sets this bit when it detects no activity for 3ms. The USB device enters Suspend mode.

- **RXRSM: UDP Resume Interrupt Status**

0 = No UDP Resume Interrupt pending.

1 =UDP Resume Interrupt has been raised.

The USB device sets this bit when a UDP resume signal is detected at its port.

After reset, the state of this bit is undefined, the application must clear this bit by setting the RXRSM flag in the UDP_ ICR register.

- **EXTRSM: External Resume Interrupt Status**

0 = No External Resume Interrupt pending.

1 = External Resume Interrupt has been raised.

This interrupt is raised when, in suspend mode, an asynchronous rising edge on the send_resume is detected.

If RMWUPE = 1, a resume state is sent in the USB bus.

- **SOFINT: Start of Frame Interrupt Status**

0 = No Start of Frame Interrupt pending.

1 = Start of Frame Interrupt has been raised.

This interrupt is raised each time a SOF token has been detected. It can be used as a synchronization signal by using isochronous endpoints.

- **ENDBUSRES: End of BUS Reset Interrupt Status**

0 = No End of Bus Reset Interrupt pending.

1 = End of Bus Reset Interrupt has been raised.

This interrupt is raised at the end of a UDP reset sequence. The USB device must prepare to receive requests on the endpoint 0. The host starts the enumeration, then performs the configuration.

- **WAKEUP: UDP Resume Interrupt Status**

0 = No Wakeup Interrupt pending.

1 = A Wakeup Interrupt (USB Host Sent a RESUME or RESET) occurred since the last clear.

After reset the state of this bit is undefined, the application must clear this bit by setting the WAKEUP flag in the UDP_ ICR register.

36.6.8 UDP Interrupt Clear Register

Register Name: UDP_ ICR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	WAKEUP	ENDBURSES	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **RXSUSP: Clear UDP Suspend Interrupt**

0 = No effect.

1 = Clears UDP Suspend Interrupt.

- **RXRSM: Clear UDP Resume Interrupt**

0 = No effect.

1 = Clears UDP Resume Interrupt.

- **EXTRSM: Clear External Resume Interrupt**

0 = No effect.

1 = Clears External Resume Interrupt.

- **SOFINT: Clear Start Of Frame Interrupt**

0 = No effect.

1 = Clears Start Of Frame Interrupt.

- **ENDBURSES: Clear End of Bus Reset Interrupt**

0 = No effect.

1 = Clears End of Bus Reset Interrupt.

- **WAKEUP: Clear Wakeup Interrupt**

0 = No effect.

1 = Clears Wakeup Interrupt.

36.6.9 UDP Reset Endpoint Register

Register Name: UDP_ RST_EP

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	EP5	EP4	EP3	EP2	EP1	EP0

- **EP0: Reset Endpoint 0**
- **EP1: Reset Endpoint 1**
- **EP2: Reset Endpoint 2**
- **EP3: Reset Endpoint 3**
- **EP4: Reset Endpoint 4**
- **EP5: Reset Endpoint 5**

This flag is used to reset the FIFO associated with the endpoint and the bit RXBYTECOUNT in the register UDP_CSRx.It also resets the data toggle to DATA0. It is useful after removing a HALT condition on a BULK endpoint. Refer to Chapter 5.8.5 in the *USB Serial Bus Specification, Rev.2.0*.

Warning: This flag must be cleared at the end of the reset. It does not clear UDP_ CSRx flags.

0 = No reset.

1 = Forces the corresponding endpoint FIF0 pointers to 0, therefore RXBYTECNT field is read at 0 in UDP_ CSRx register.

36.6.10 UDP Endpoint Control and Status Register

Register Name: UDP_ CSRx [x = 0..5]

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	RXBYTECNT		

23	22	21	20	19	18	17	16
RXBYTECNT							

15	14	13	12	11	10	9	8
EPEDS	–	–	–	DTGLE	EPTYPE		

7	6	5	4	3	2	1	0
DIR	RX_DATA_ BK1	FORCE STALL	TXPKTRDY	STALLSENT ISOERROR	RXSETUP	RX_DATA_ BK0	TXCOMP

- **TXCOMP: Generates an IN packet with data previously written in the DPR**

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware):

0 = Clear the flag, clear the interrupt.

1 = No effect.

Read (Set by the USB peripheral):

0 = Data IN transaction has not been acknowledged by the Host.

1 = Data IN transaction is achieved, acknowledged by the Host.

After having issued a Data IN transaction setting TXPKTRDY, the device firmware waits for TXCOMP to be sure that the host has acknowledged the transaction.

- **RX_DATA_BK0: Receive Data Bank 0**

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware):

0 = Notify USB peripheral device that data have been read in the FIFO's Bank 0.

1 = No effect.

Read (Set by the USB peripheral):

0 = No data packet has been received in the FIFO's Bank 0

1 = A data packet has been received, it has been stored in the FIFO's Bank 0.

When the device firmware has polled this bit or has been interrupted by this signal, it must transfer data from the FIFO to the microcontroller memory. The number of bytes received is available in RXBYTCENT field. Bank 0 FIFO values are read through the UDP_ FDRx register. Once a transfer is done, the device firmware must release Bank 0 to the USB peripheral device by clearing RX_DATA_BK0.

- **RXSETUP: Sends STALL to the Host (Control Endpoints)**

This flag generates an interrupt while it is set to one.

Read:

0 = No setup packet available.

1 = A setup data packet has been sent by the host and is available in the FIFO.

Write:

0 = Device firmware notifies the USB peripheral device that it has read the setup data in the FIFO.

1 = No effect.

This flag is used to notify the USB device firmware that a valid Setup data packet has been sent by the host and successfully received by the USB device. The USB device firmware may transfer Setup data from the FIFO by reading the UDP_ FDRx register to the microcontroller memory. Once a transfer has been done, RXSETUP must be cleared by the device firmware.

Ensuing Data OUT transaction is not accepted while RXSETUP is set.

- **STALLSENT: Stall Sent (Control, Bulk Interrupt Endpoints) / ISOERROR (Isochronous Endpoints)**

This flag generates an interrupt while it is set to one.

STALLSENT: This ends a STALL handshake.

Read:

0 = The host has not acknowledged a STALL.

1 = Host has acknowledged the stall.

Write:

0 = Resets the STALLSENT flag, clears the interrupt.

1 = No effect.

This is mandatory for the device firmware to clear this flag. Otherwise the interrupt remains.

Refer to chapters 8.4.5 and 9.4.5 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the STALL handshake.

ISOERROR: A CRC error has been detected in an isochronous transfer.

Read:

0 = No error in the previous isochronous transfer.

1 = CRC error has been detected, data available in the FIFO are corrupted.

Write:

0 = Resets the ISOERROR flag, clears the interrupt.

1 = No effect.

- **TXPKTRDY: Transmit Packet Ready**

This flag is cleared by the USB device.

This flag is set by the USB device firmware.

Read:

0 = Data values can be written in the FIFO.

1 = Data values can not be written in the FIFO.

Write:

0 = No effect.

1 = A new data payload is has been written in the FIFO by the firmware and is ready to be sent.

This flag is used to generate a Data IN transaction (device to host). Device firmware checks that it can write a data payload in the FIFO, checking that TXPKTRDY is cleared. Transfer to the FIFO is done by writing in the UDP_ FDRx register. Once the data payload has been transferred to the FIFO, the firmware notifies the USB device setting TXPKTRDY to one. USB bus transactions can start. TXCOMP is set once the data payload has been received by the host.

- **FORCESTALL: Force Stall (used by Control, Bulk and Isochronous Endpoints)**

Write-only

0 = No effect.

1 = Sends STALL to the host.

Refer to chapters 8.4.5 and 9.4.5 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the STALL handshake.

Control endpoints: During the data stage and status stage, this indicates that the microcontroller cannot complete the request.

Bulk and interrupt endpoints: Notifies the host that the endpoint is halted.

The host acknowledges the STALL, device firmware is notified by the STALLSENT flag.

- **RX_DATA_BK1: Receive Data Bank 1 (only used by endpoints with ping-pong attributes)**

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware):

0 = Notifies USB device that data have been read in the FIFO's Bank 1.

1 = No effect.

Read (Set by the USB peripheral):

0 = No data packet has been received in the FIFO's Bank 1.

1 = A data packet has been received, it has been stored in FIFO's Bank 1.

When the device firmware has polled this bit or has been interrupted by this signal, it must transfer data from the FIFO to microcontroller memory. The number of bytes received is available in RXBYTECNT field. Bank 1 FIFO values are read through UDP_ FDRx register. Once a transfer is done, the device firmware must release Bank 1 to the USB device by clearing RX_DATA_BK1.

- **DIR: Transfer Direction (only available for control endpoints)**

Read/Write

0 = Allows Data OUT transactions in the control data stage.

1 = Enables Data IN transactions in the control data stage.

Refer to Chapter 8.5.3 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the control data stage.

This bit must be set before UDP_ CSRx/RXSETUP is cleared at the end of the setup stage. According to the request sent in the setup data packet, the data stage is either a device to host (DIR = 1) or host to device (DIR = 0) data transfer. It is not necessary to check this bit to reverse direction for the status stage.

- **EPTYPE[2:0]: Endpoint Type**

Read/Write

000	Control
001	Isochronous OUT
101	Isochronous IN
010	Bulk OUT
110	Bulk IN
011	Interrupt OUT
111	Interrupt IN

- **DTGLE: Data Toggle**

Read-only

0 = Identifies DATA0 packet.

1 = Identifies DATA1 packet.

Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on DATA0, DATA1 packet definitions.

- **EPEDS: Endpoint Enable Disable**

Read:

0 = Endpoint disabled.

1 = Endpoint enabled.

Write:

0 = Disables endpoint.

1 = Enables endpoint.

- **RXBYTECNT[10:0]: Number of Bytes Available in the FIFO**

Read-only

When the host sends a data packet to the device, the USB device stores the data in the FIFO and notifies the microcontroller. The microcontroller can load the data from the FIFO by reading RXBYTECENT bytes in the UDP_ FDRx register.

36.6.11 UDP FIFO Data Register

Register Name: UDP_ FDRx [x = 0..5]

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FIFO_DATA							

• **FIFO_DATA[7:0]: FIFO Data Value**

The microcontroller can push or pop values in the FIFO through this register.

RXBYTECNT in the corresponding UDP_ CSRx register is the number of bytes to be read from the FIFO (sent by the host).

The maximum number of bytes to write is fixed by the Max Packet Size in the Standard Endpoint Descriptor. It can not be more than the physical memory size associated to the endpoint. Refer to the *Universal Serial Bus Specification, Rev. 2.0* for more information.

36.6.12 UDP Transceiver Control Register

Register Name: UDP_ TXVC

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	TXVDIS
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	–

• **TXVDIS: Transceiver Disable**

When UDP is disabled, power consumption can be reduced significantly by disabling the embedded transceiver. This can be done by setting TXVDIS field.

To enable the transceiver, TXVDIS must be cleared.

37. LCD Controller (LCDC)

37.1 Description

The LCD Controller consists of logic for transferring LCD image data from an external display buffer to an LCD module with integrated common and segment drivers.

The LCD Controller supports single and double scan monochrome and color passive STN LCD modules and single scan active TFT LCD modules. On monochrome STN displays, up to 16 gray shades are supported using a time-based dithering algorithm and Frame Rate Control (FRC) method. This method is also used in color STN displays to generate up to 4096 colors.

The LCD Controller has a display input buffer (FIFO) to allow a flexible connection of the external AHB master interface, and a lookup table to allow palletized display configurations.

The LCD Controller is programmable in order to support many different requirements such as resolutions up to 2048 x 2048; pixel depth (1, 2, 4, 8, 16, 24 bits per pixel); data line width (4, 8, 16 or 24 bits) and interface timing.

The LCD Controller is connected to the ARM Advanced High Performance Bus (AHB) as a master for reading pixel data. However, the LCD Controller interfaces with the AHB as a slave in order to configure its registers.

37.2 Block Diagram

Figure 37-1. LCD Macrocell Block Diagram

AHB MASTER
AHB SLAVE
DMA Controller
AHB IF
CFG
SPLIT
AHB SLAVE
DMA Data
Dvalid
Dvalid
CH-U
CH-L
CTRL
Upper Push
Lower Push
Input Interface
LCD Controller Core
FIFO
Configuration IF
CFG
AHB SLAVE
AHB
Clock Domain
SERIALIZER
LCDC Core
Clock Domain
DATAPATH
PALETTE
LUT Mem Interface
LUT Mem Interface
FIFO Mem Interface
DITHERING
Control Interface
FIFO
MEM
LUT
MEM
OUTPUT
SHIFTER
Timegen
LCDD
DISPLAY IF
Control signals
Display
PWM
DISPLAY IF

37.3 I/O Lines Description

Table 37-1. I/O Lines Description

Name	Description	Type
LCD_CC	Contrast control signal	Output
LCD_HSYNC	Line synchronous signal (STN) or Horizontal synchronous signal (TFT)	Output
LCD_PCLK	LCD clock signal (STN/TFT)	Output
LCD_VSYNC	Frame synchronous signal (STN) or Vertical synchronization signal (TFT)	Output
LCD_DVAL	STN AC bias signal for the driver or Data enable signal (TFT)	Output
LCD_MOD	LCD Modulation signal	Output
LCD_PWR	LCD panel Power enable control signal	Output
LCDD[23:0]	LCD Data Bus output	Output

37.4 Product Dependencies

37.4.1 I/O Lines

The pins used for interfacing the LCD Controller may be multiplexed with PIO lines. The programmer must first program the PIO Controller to assign the pins to their peripheral function. If I/O lines of the LCD Controller are not used by the application, they can be used for other purposes by the PIO Controller.

37.4.2 Power Management

The LCD Controller is not continuously clocked. The user must first enable the LCD Controller clock in the Power Management Controller before using it. As the LCD Controller is on the AHB bus, the clock is enabled by setting the HCLK bit in the PMC_SCER register.

37.4.3 Interrupt Sources

The LCD Controller interrupt line is connected to one of the internal sources of the Advanced Interrupt Controller. Using the LCD Controller interrupt requires prior programming of the AIC.

37.5 Functional Description

The LCD Controller consists of two main blocks (Figure 37-1 on page 560), the DMA controller and the LCD controller core (LCDC core). The DMA controller reads the display data from an external memory through a AHB master interface. The LCD controller core formats the display data. The LCD controller core continuously pumps the pixel data into the LCD module via the LCD data bus (LCDD[23:0]); this bus is timed by the LCD_PCLK, LCD_DVAL, LCD_HSYNC, and LCD_VSYNC signals.

37.5.1 DMA Controller

37.5.1.1 Configuration Block

The configuration block is a set of programmable registers that are used to configure the DMA controller operation. These registers are written via the AHB slave interface. Only word access is allowed.

For details on the configuration registers, see "LCD Controller (LCDC) User Interface" on page 585.

37.5.1.2 AHB Interface

This block generates the AHB transactions. It generates undefined-length incrementing bursts as well as 4- ,8- or 16-beat incrementing bursts. The size of the transfer can be configured in the BRSTLN field of the DMAFRMCFG register. For details on this register, see ”DMA Frame Configuration Register” on page 590.

37.5.1.3 Channel-U

This block stores the base address and the number of words transferred for this channel (frame in single scan mode and Upper Panel in dual scan mode) since the beginning of the frame. It also generates the end of frame signal.

It has two pointers, the base address and the number of words to transfer. When the module receives a new_frame signal, it reloads the number of words to transfer pointer with the size of the frame/panel. When the module receives the new_frame signal, it also reloads the base address with the base address programmed by the host.

The size of the frame/panel can be programmed in the FRMSIZE field of the DMAFRMCFG Register. This size is calculated as follows:

- In TFT mode:

$$\text{Frame_size} = \left\lceil \frac{\text{Display_size} \times \text{Bpp}}{32} \right\rceil$$

- In STN Monochrome mode:

$$\text{Frame_size} = \left\lceil \frac{(\text{LINEVAL} + 1) \times (\text{HOZVAL} + 1) \times \text{E_ifwidth} \times \text{Bpp}}{32} \right\rceil$$

- In STN Color mode:

$$\text{Frame_size} = \left\lceil \frac{(\text{LINEVAL} + 1)\left(\frac{(\text{HOZVAL} + 1) \times \text{E_ifwidth}}{3}\right)\text{Bpp}}{32} \right\rceil$$

where:

- LINEVAL is the value of the LINEVAL field of the LCDFRMCFG register of LCD Controller
- HOZVAL is the value of the HOZVAL field of the LCDFRMCFG register of the LCD Controller
- E_ifwidth is the number of data bits in the LCD interface for each panel
- Bpp is the bits per pixel configuration

37.5.1.4 Channel-L

This block has the same functionality as Channel-U, but for the Lower Panel in dual scan mode only.

37.5.1.5 Control

This block receives the request signals from the LCDC core and generates the requests for the channels.

37.5.2 LCD Controller Core

37.5.2.1 Configuration Block

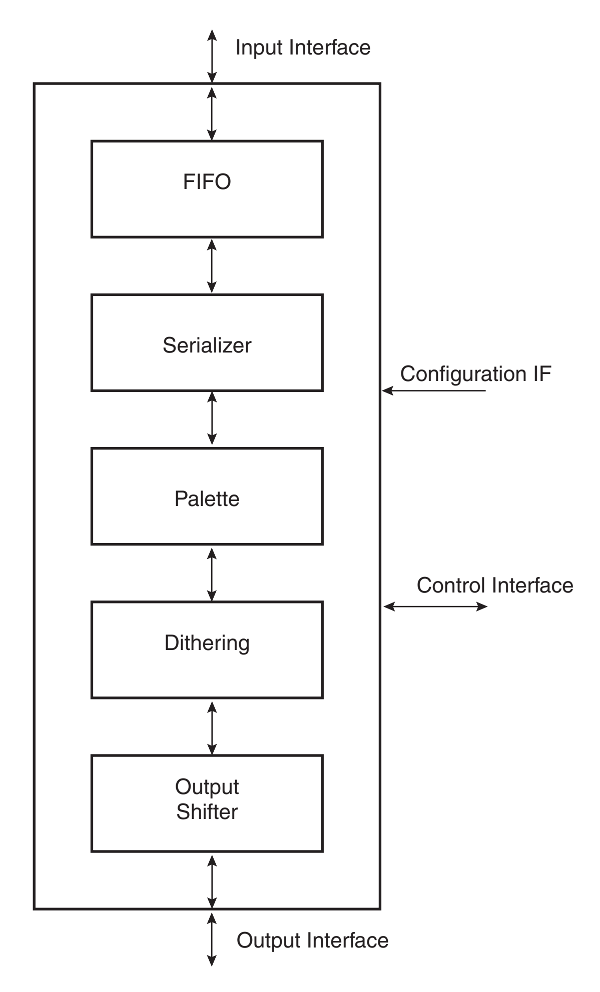
The configuration block is a set of programmable registers that are used to configure the LCDC core operation. These registers are written via the AHB slave interface. Only word access is allowed.

The description of the configuration registers can be found in ”LCD Controller (LCDC) User Interface” on page 585.

37.5.2.2 Datapath

The datapath block contains five submodules: FIFO, Serializer, Palette, Dithering and Shifter. The structure of the datapath is shown in Figure 37-2.

Figure 37-2. Datapath Structure



This module transforms the data read from the memory into a format according to the LCD module used. It has four different interfaces: the input interface, the output interface, the configuration interface and the control interface.

- The input interface connects the datapath with the DMA controller. It is a dual FIFO interface with a data bus and two push lines that are used by the DMA controller to fill the FIFOs.

- The output interface is a 24-bit data bus. The configuration of this interface depends on the type of LCD used (TFT or STN, Single or Dual Scan, 4-bit, 8-bit, 16-bit or 24-bit interface).
- The configuration interface connects the datapath with the configuration block. It is used to select between the different datapath configurations.
- The control interface connects the datapath with the timing generation block. The main control signal is the data-request signal, used by the timing generation module to request new data from the datapath.

The datapath can be characterized by two parameters: initial_latency and cycles_per_data. The parameter initial_latency is defined as the number of LCDC Core Clock cycles until the first data is available at the output of the datapath. The parameter cycles_per_data is the minimum number of LCDC Core clock cycles between two consecutive data at the output interface.

These parameters are different for the different configurations of the LCD Controller and are shown in Table 37-2.

Table 37-2. Datapath Parameters

Configuration				
DISTYPE	**SCAN**	**IFWIDTH**	**initial_latency**	**cycles_per_data**
TFT			9	1
STN Mono	Single	4	13	4
STN Mono	Single	8	17	8
STN Mono	Dual	8	17	8
STN Mono	Dual	16	25	16
STN Color	Single	4	11	2
STN Color	Single	8	12	3
STN Color	Dual	8	14	4
STN Color	Dual	16	15	6

FIFO

The FIFO block buffers the input data read by the DMA module. It contains two input FIFOs to be used in Dual Scan configuration that are configured as a single FIFO when used in single scan configuration.

The size of the FIFOs allows a wide range of architectures to be supported.

The upper threshold of the FIFOs can be configured in the FIFOTH field of the LCDFIFO register. The LCDC core will request a DMA transfer when the number of words in each FIFO is less than FIFOTH words. To avoid overwriting in the FIFO and to maximize the FIFO utilization, the FIFOTH should be programmed with:

FIFOTH = 512 words - (2 x DMA_burst_length + 3)

where:

- 512 words is the effective size of the FIFO. It is the total FIFO memory size in single scan mode and half that size in dual scan mode.
- DMA_burst_length is the burst length of the transfers made by the DMA

Serializer

This block serializes the data read from memory. It reads words from the FIFO and outputs pixels (1 bit, 2 bits, 4 bits, 8 bits, 16 bits or 24 bits wide) depending on the format specified in the PIXELSIZE field of the LCDCON2 register. It also adapts the memory-ordering format. Both big-endian and little-endian formats are supported. They are configured in the MEMOR field of the LCDCON2 register.

The organization of the pixel data in the memory depends on the configuration and is shown in Table 37-3 and Table 37-4.

Table 37-3. Little Endian Memory Organization

Mem Addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 1bpp	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 2bpp	15		14		13		12		11		10		9		8		7		6		5		4		3		2		1		0	
Pixel 4bpp	7				6				5				4				3				2				1				0			
Pixel 8bpp	3								2								1								0							
Pixel 16bpp	1																0															
Pixel 24bpp	1								0																							
Pixel 24bpp	2																1															
Pixel 24bpp	3																								2							
Pixel 24bpp	5								4																							

Table 37-4. Big Endian Memory Organization

Mem Addr	0x3								0x2								0x1								0x0							
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 1bpp	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pixel 2bpp	0		1		2		3		4		5		6		7		8		9		10		11		12		13		14		15	
Pixel 4bpp	0				1				2				3				4				5				6				7			
Pixel 8bpp	0								1								2								3							
Pixel 16bpp	0																1															
Pixel 24bpp	0																								1							
Pixel 24bpp	1																2															
Pixel 24bpp	2								3																							
Pixel 24bpp	4																								5							

Palette

This block is used to generate the pixel gray or color information in palletized configurations. The different modes with the palletized/non-palletized configuration can be found in Table 37-5. In these modes, 1, 2, 4 or 8 input bits index an entry in the lookup table. The corresponding entry in the lookup table contains the color or gray shade information for the pixel.

Table 37-5. Palette Configurations

Configuration		
DISTYPE	**PIXELSIZE**	**Palette**
TFT	1, 2, 4, 8	Palletized
TFT	16, 24	Non-palletized
STN Mono	1, 2	Palletized
STN Mono	4	Non-palletized
STN Color	1, 2, 4, 8	Palletized
STN Color	16	Non-palletized

The lookup table can be accessed by the host in R/W mode to allow the host to program and check the values stored in the palette. It is mapped in the LCD controller configuration memory map. The LUT is mapped as 16-bit half-words aligned at word boundaries, only word write access is allowed (the 16 MSB of the bus are not used). For the detailed memory map, see Table 37-12 on page 585.

The lookup table contains 256 16-bit wide entries. The 256 entries are chosen by the programmer from the 2^{16} possible combinations.

For the structure of each LUT entry, see Table 37-6.

Table 37-6. Lookup Table Structure in the Memory

Address	Data Output [15:0]			
00	Intensity_bit_0	Blue_value_0[4:0]	Green_value_0[4:0]	Red_value_0[4:0]
01	Intensity_bit_1	Blue_value_1[4:0]	Green_value_1[4:0]	Red_value_1[4:0]
...				
FE	Intensity_bit_254	Blue_value_254[4:0]	Green_value_254[4:0]	Red_value_254[4:0]
FF	Intensity_bit_255	Blue_value_255[4:0]	Green_value_255[4:0]	Red_value_255[4:0]

In STN Monochrome, only the four most significant bits of the red value are used (16 gray shades). In STN Color, only the four most significant bits of the blue, green and red value are used (4096 colors).

In TFT mode, all the bits in the blue, green and red values are used (32768 colors). In this mode, there is also a common intensity bit that can be used to double the possible colors. This bit is the least significant bit of each color component in the LCDD interface (LCDD[18], LCDD[10], LCDD[2]). The LCDD unused bits are tied to 0 when TFT palletized configurations are used (LCDD[17:16], LCDD[9:8], LCDD[1:0]).

Dithering

The dithering block is used to generate the shades of gray or color when the LCD Controller is used with an STN LCD Module. It uses a time-based dithering algorithm and Frame Rate Control method.

The Frame Rate Control varies the duty cycle for which a given pixel is turned on, giving the display an appearance of multiple shades. In order to reduce the flicker noise caused by turning on and off adjacent pixels at the same time, a time-based dithering algorithm is used to vary the pattern of adjacent pixels every frame. This algorithm is expressed in terms of Dithering Pattern registers (DP_i) and considers not only the pixel gray level number, but also its horizontal coordinate.

Table 37-7 shows the correspondences between the gray levels and the duty cycle.

Table 37-7. Dithering Duty Cycle

Gray Level	Duty Cycle	Pattern Register
15	1	-
14	6/7	DP6_7
13	4/5	DP4_5
12	3/4	DP3_4
11	5/7	DP5_7
10	2/3	DP2_3
9	3/5	DP3_5
8	4/7	DP4_7
7	1/2	~DP1_2
6	3/7	~DP4_7
5	2/5	~DP3_5
4	1/3	~DP2_3
3	1/4	~DP3_4
2	1/5	~DP4_5
1	1/7	~DP6_7
0	0	-

The duty cycles for gray levels 0 and 15 are 0 and 1, respectively.

The same DP_i register can be used for the pairs for which the sum of duty cycles is 1 (e.g., 1/7 and 6/7). The dithering pattern for the first pair member is the inversion of the one for the second.

The DP_i registers contain a series of 4-bit patterns. The $(3\text{-}m)^{th}$ bit of the pattern determines if a pixel with horizontal coordinate $x = 4n + m$ (n is an integer and m ranges from 0 to 3) should be turned on or off in the current frame. The operation is shown by the examples below.

Consider the pixels a, b, c and d with the horizontal coordinates 4*n+0, 4*n+1, 4*n+2 and 4*n+3, respectively. The four pixels should be displayed in gray level 9 (duty cycle 3/5) so the register used is DP3_5 =”1010 0101 1010 0101 1111”.

The output sequence obtained in the data output for monochrome mode is shown in Table 37-8.

Table 37-8. Dithering Algorithm for Monochrome Mode

Frame Number	Pattern	Pixel a	Pixel b	Pixel c	Pixel d
N	1010	ON	OFF	ON	OFF
N+1	0101	OFF	ON	OFF	ON
N+2	1010	ON	OFF	ON	OFF
N+3	0101	OFF	ON	OFF	ON
N+4	1111	ON	ON	ON	ON
N+5	1010	ON	OFF	ON	OFF
N+6	0101	OFF	ON	OFF	ON
N+7	1010	ON	OFF	ON	OFF
...	...	...	...	...	...

Consider now color display mode and two pixels p0 and p1 with the horizontal coordinates 4*n+0, and 4*n+1. A color pixel is composed of three components: {R, G, B}. Pixel p0 will be displayed sending the color components {R0, G0, B0} to the display. Pixel p1 will be displayed sending the color components {R1, G1, B1}. Suppose that the data read from memory and mapped to the lookup tables corresponds to shade level 10 for the three color components of both pixels, with the dithering pattern to apply to all of them being DP2_3 = “1101 1011 0110”. Table 37-9 shows the output sequence in the data output bus for single scan configurations. (In Dual Scan Configuration, each panel data bus acts like in the equivalent single scan configuration.)

Table 37-9. Dithering Algorithm for Color Mode

Frame	Signal	Shadow Level	Bit used	Dithering Pattern	4-bit LCDD	8-bit LCDD	Output
N	red_data_0	1010	3	1101	LCDD[3]	LCDD[7]	R0
N	green_data_0	1010	2	1101	LCDD[2]	LCDD[6]	G0
N	blue_data_0	1010	1	1101	LCDD[1]	LCDD[5]	b0
N	red_data_1	1010	0	1101	LCDD[0]	LCDD[4]	R1
N	green_data_1	1010	3	1101	LCDD[3]	LCDD[3]	G1
N	blue_data_1	1010	2	1101	LCDD[2]	LCDD[2]	B1
…	…	…	…	…	…	…	…
N+1	red_data_0	1010	3	1011	LCDD[3]	LCDD[7]	R0
N+1	green_data_0	1010	2	1011	LCDD[2]	LCDD[6]	g0
N+1	blue_data_0	1010	1	1011	LCDD[1]	LCDD[5]	B0
N+1	red_data_1	1010	0	1011	LCDD[0]	LCDD[4]	R1
N+1	green_data_1	1010	3	1011	LCDD[3]	LCDD[3]	G1
N+1	blue_data_1	1010	2	1011	LCDD[2]	LCDD[2]	b1
…	…	…	…	…	…	…	…

Table 37-9. Dithering Algorithm for Color Mode (Continued)

Frame	Signal	Shadow Level	Bit used	Dithering Pattern	4-bit LCDD	8-bit LCDD	Output
N+2	red_data_0	1010	3	0110	LCDD[3]	LCDD[7]	r0
N+2	green_data_0	1010	2	0110	LCDD[2]	LCDD[6]	G0
N+2	blue_data_0	1010	1	0110	LCDD[1]	LCDD[5]	B0
N+2	red_data_1	1010	0	0110	LCDD[0]	LCDD[4]	r1
N+2	green_data_1	1010	3	0110	LCDD[3]	LCDD[3]	g1
N+2	blue_data_1	1010	2	0110	LCDD[2]	LCDD[2]	B1
...	...	...	...	...	...	...	...

Note: Ri = red pixel component ON. Gi = green pixel component ON. Bi = blue pixel component ON. ri = red pixel component OFF. gi = green pixel component OFF. bi = blue pixel component OFF.

Shifter

The FIFO, Serializer, Palette and Dithering modules process one pixel at a time in monochrome mode and three sub-pixels at a time in color mode (R,G,B components). This module packs the data according to the output interface. This interface can be programmed in the DISTYPE, SCANMOD, and IFWIDTH fields of the LDCCON3 register.

The DISTYPE field selects between TFT, STN monochrome and STN color display. The SCANMODE field selects between single and dual scan modes; in TFT mode, only single scan is supported. The IFWIDTH field configures the width of the interface in STN mode: 4-bit (in single scan mode only), 8-bit and 16-bit (in dual scan mode only).

For a more detailed description of the fields, see ”LCD Controller (LCDC) User Interface” on page 585.

For a more detailed description of the LCD Interface, see ”LCD Interface” on page 575.

37.5.2.3 Timegen

The time generator block generates the control signals LCD_PCLK, LCD_HSYNC, LCD_VSYNC, LCD_DVAL, and LCD_MODE, used by the LCD module. This block is programmable in order to support different types of LCD modules and obtain the output clock signals, which are derived from the LCDC Core clock.

The LCD_MODE signal provides an AC signal for the display. It is used by the LCD to alternate the polarity of the row and column voltages used to turn the pixels on and off. This prevents the liquid crystal from degradation. It can be configured to toggle every frame (bit MMODE = 0 in LCDMVAL register) or to toggle every programmable number of LCD_HSYNC pulses (bit MMODE = 1, number of pulses defined in MVAL field of LCDMVAL register).

$$f_{LCD_MODE} = \frac{f_{LCD_HSYNC}}{2 \times (MVAL + 1)}$$

Figure 37-3 and Figure 37-4 on page 570 show the timing of LCD_MODE in both configurations.



Figure 37-3. Full Frame Timing, MMODE=1, MVAL=1

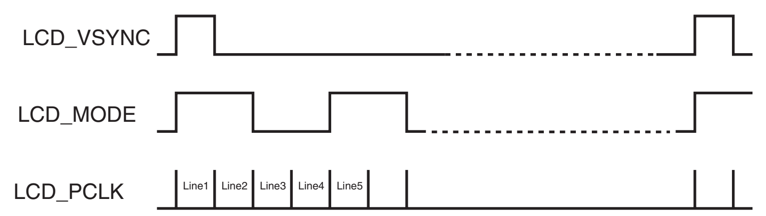
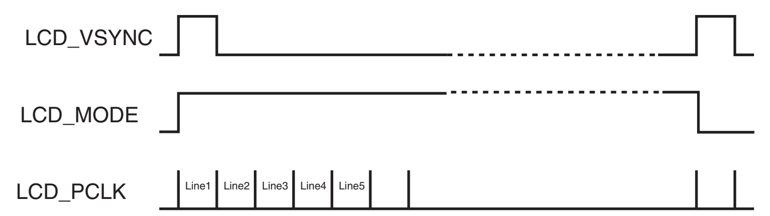


Figure 37-4. Full Frame Timing, MMODE=0



The LCD_PCLK signal is used to clock the data into the LCD drivers' shift register. The data is sent through LCDD[23:0] synchronized by default with lcd_pclk falling edge (rising edge can be selected). The CLKVAL field of LCDCON1 register controls the rate of this signal. The divisor can also be bypassed with the BYPASS bit in the LCDCON1 register. In this case, the rate of LCD_PCLK is equal to the frequency of the LCDC Core clock. The minimum period of the LCD_PCLK signal depends on the configuration. This information can be found in Table 37-10.

$$f_{LCD_PCLK} = \frac{f_{LCDC_clock}}{2 \times CLKVAL}$$

The LCD_PCLK signal has two different timings that are selected with the CLKMOD field of the LCDCON2 register:

- Always Active (used with TFT LCD Modules)
- Active only when data is available (used with STN LCD Modules)

Table 37-10. Minimum LCD_PCLK Period in LCDC Core Clock Cycles

Configuration			
DISTYPE	**SCAN**	**IFWIDTH**	**LCD_PCLK Period**
TFT			1
STN Mono	Single	4	4
STN Mono	Single	8	8
STN Mono	Dual	8	8
STN Mono	Dual	16	16
STN Color	Single	4	2

Table 37-10. Minimum LCD_PCLK Period in LCDC Core Clock Cycles

Configuration			
DISTYPE	**SCAN**	**IFWIDTH**	**LCD_PCLK Period**
STN Color	Single	8	2
STN Color	Dual	8	4
STN Color	Dual	16	6

The LCD_DVAL signal indicates valid data in the LCD Interface.

After each horizontal line of data has been shifted into the LCD, the LCD_HSYNC is asserted to cause the line to be displayed on the panel.

The following timing parameters can be configured:

- Vertical to Horizontal Delay (VHDLY): The delay between begin_of_line and the generation of LCD_HSYNC is configurable in the VHDLY field of the LCDTIM1 register. The delay is equal to (VHDLY+1) LCD_PCLK cycles.
- Horizontal Pulse Width (HPW): The LCD_HSYNC pulse width is configurable in HPW field of LCDTIM2 register. The width is equal to (HPW + 1) LCD_PCLK cycles.
- Horizontal Back Porch (HBP): The delay between the LCD_HSYNC falling edge and the first LCD_PCLK rising edge with valid data at the LCD Interface is configurable in the HBP field of the LCDTIM2 register. The delay is equal to (HBP+1) lcd_pclk cycles.
- Horizontal Front Porch (HFP): The delay between end of valid data and the end of the line is configurable in the HFP field of the LCDTIM2 register. The delay is equal to (HFP+1) LCD_PCLK cycles.

There is a limitation in the minimum values of VHDLY, HPW and HBP parameters imposed by the initial latency of the datapath. The total delay in LCDC clock cycles must be higher than or equal to the latency column in Table 37-2 on page 564. This limitation is given by the following formula:

Equation 1

$$(\mathrm{VHDLY} + \mathrm{HPW} + \mathrm{HBP} + 3) \times \mathrm{PCLK_PERIOD} \geq \mathrm{DPATH_LATENCY}$$

where:

- VHDLY, HPW, HBP are the value of the fields of LCDTIM1 and LCDTIM2 registers
- PCLK_PERIOD is the period of lcd_pclk signal measured in LCDC Clock cycles
- DPATH_LATENCY is the datapath latency of the configuration, given in Table 37-2 on page 564

The LCD_VSYNC is asserted once per frame. This signal is asserted to cause the LCD's line pointer to start over at the top of the display. The timing of this signal depends on the type of LCD: STN or TFT LCD.

In STN mode, the high phase corresponds to the complete first line of the frame. In STN mode, this signal is synchronized with the first active LCD_PCLK rising edge in a line.

In TFT mode, the high phase of this signal starts at the beginning of the first line. The following timing parameters can be selected:

- Vertical Pulse Width (VPW): LCD_VSYNC pulse width is configurable in VPW field of the LCDTIM1 register. The pulse width is equal to (VPW+1) lines.

- Vertical Back Porch: Number of inactive lines at the beginning of the frame is configurable in VBP field of LCDTIM1 register. The number of inactive lines is equal to VBP. This field should be programmed with 0 in STN Mode.
- Vertical Front Porch: Number of inactive lines at the end of the frame is configurable in VFP field of LCDTIM2 register. The number of inactive lines is equal to VFP. This field should be programmed with 0 in STN mode.

There are two other parameters to configure in this module, the HOZVAL and the LINEVAL fields of the LCDFRMCFG:

- HOZVAL configures the number of active LCD_PCLK cycles in each line. The number of active cycles in each line is equal to (HOZVAL+1) cycles. The minimum value of this parameter is 1.
- LINEVAL configures the number of active lines per frame. This number is equal to (LINEVAL+1) lines. The minimum value of this parameter is 1.

Figure 37-5, Figure 37-6 and Figure 37-7 show the timing of LCD_MODE, LCD_PCLK, LCD_DVAL, LCD_HSYNC and LCD_VSYNC signals:

Figure 37-5. STN Panel Timing, CLKMOD 0

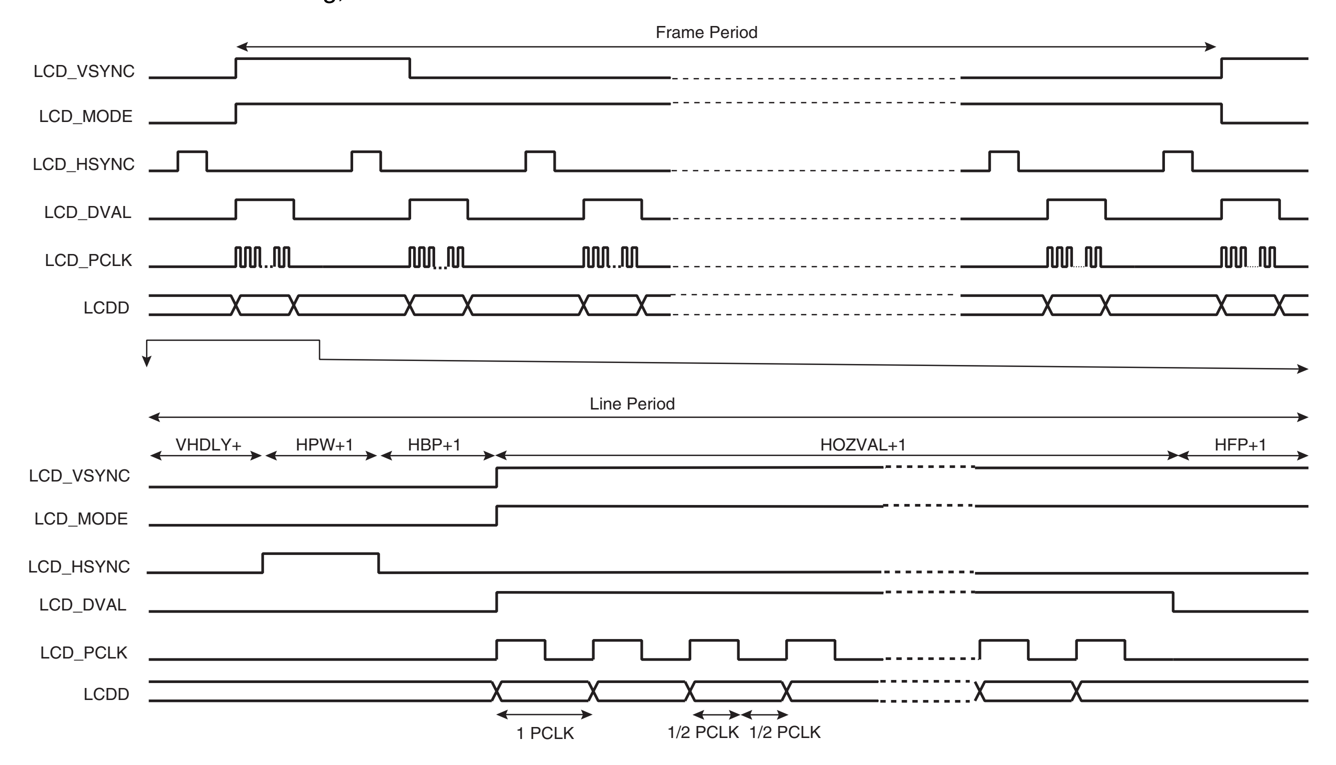




Figure 37-6. TFT Panel Timing, CLKMOD = 0, VPW = 2, VBP = 2, VFP = 1

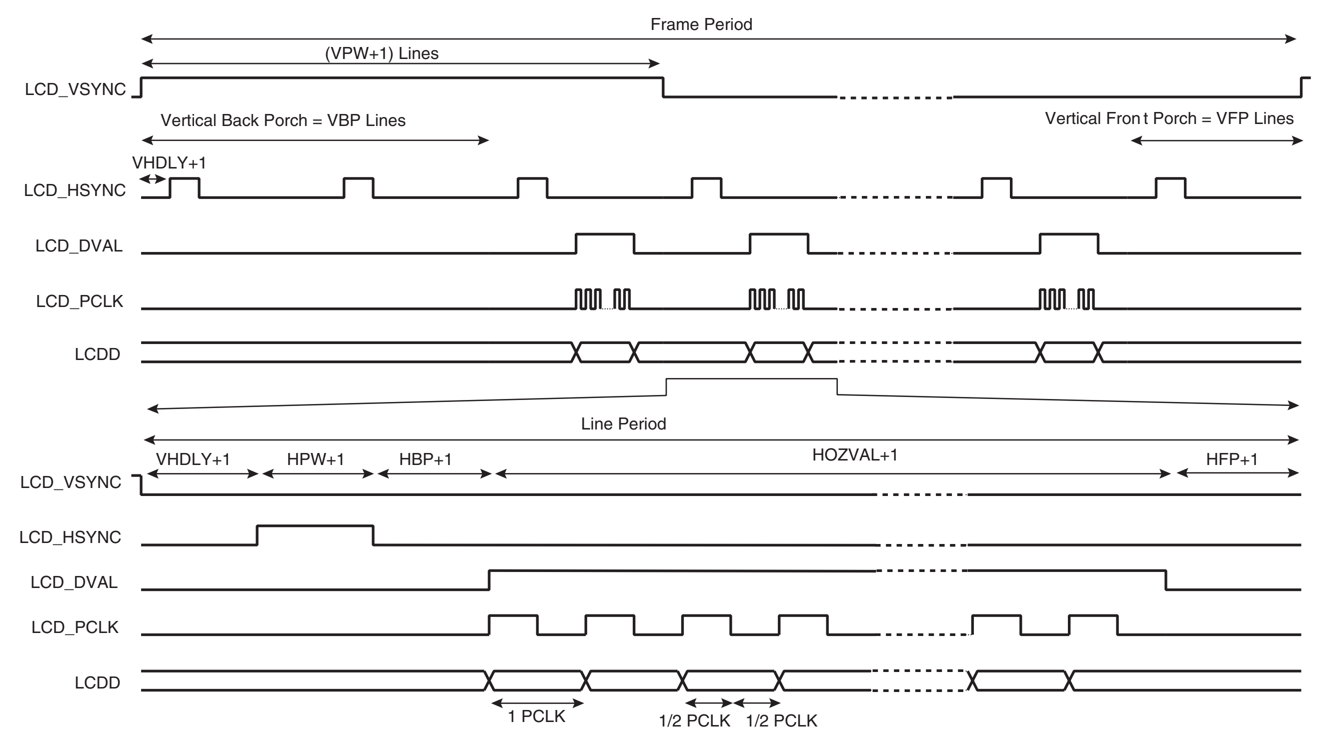
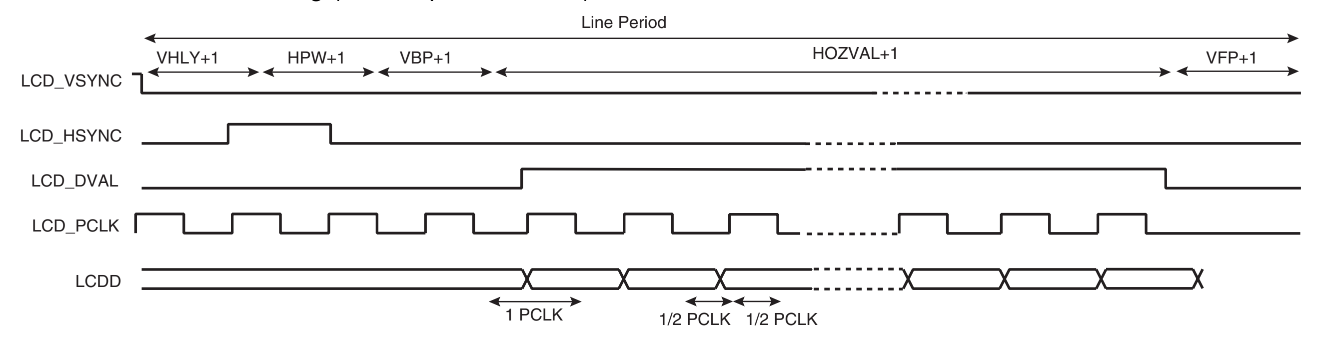


Figure 37-7. TFT Panel Timing (Line Expanded View), CLKMOD=1



Usually the LCD_FRM rate is about 70 Hz to 75 Hz. It is given by the following equation:

$$\frac{1}{f_{lcd_vsync}} = \left(\frac{VHDLY + HPW + HBP + HOZVAL + HFP + 5}{f_{lcd_pclk}}\right)(VBP + LINEVAL + VFP + 1)$$

where:

- HOZVAL determines de number of lcd_pclk cycles per line
- LINEVAL determines the number of lcd_hsync cycles per frame, according to the expressions shown below:

In STN Mode:

$$HOZVAL = \frac{Horizontal_display_size}{Number_data_lines} - 1$$

$$LINEVAL = Vertical_display_size - 1$$

In monochrome mode, Horizontal_display_size is equal to the number of horizontal pixels. The number_data_lines is equal to the number of bits of the interface in single scan mode; number_data_lines is equal to half the bits of the interface in dual scan mode.

In color mode, Horizontal_display_size equals three times the number of horizontal pixels.

In TFT Mode:

$$HOZVAL = Horizontal_display_size - 1$$

$$LINEVAL = Vertical_display_size - 1$$

The frame rate equation is used first without considering the clock periods added at the end beginning or at the end of each line to determine, approximately, the LCD_PCLK rate:

$$f_{lcd_pclk} = (HOZVAL + 5) \times (f_{lcd_vsync} \times (LINEVAL + 1))$$

With this value, the CLKVAL is fixed, as well as the corresponding LCD_PCLK rate.

Then select VHDLY, HPW and HBP according to the type of LCD used and "Equation 1" on page 571.

Finally, the frame rate is adjusted to 70 Hz - 75 Hz with the HFP value:

$$HFP = f_{pclk} \times \left[\frac{1}{f_{lcd_vsync} \times (LINEVAL + VBP + VFP + 1)}\right] - (VHDLY + VPW + VBP + HOZVAL + 5)$$

The line counting is controlled by the read-only field LINECNT of LCDCON1 register. The LINECNT field decreases by one unit at each falling edge of lcd_hsync.

37.5.2.4 Display

This block is used to configure the polarity of the data and control signals. The polarity of all clock signals can be configured by LCDCON2[12:8] register setting.

The block also generates the LCD_PWR output that can be used to turn the LCD module on and off by software. This signal is controlled by the PWRCON register and respects the number of frames configured in the GUARD_TIME field of PWRCON register (PWRCON[7:1]) between the write access to LCD_PWR field (PWRCON[0]) and the activation/deactivation of LCD_PWR output signal. The minimum value for the GUARD_TIME field is one frame. This gives the DMA Controller enough time to fill the FIFOs before the start of data transfer to the LCD.

37.5.2.5 PWM

This block generates the LCD contrast control signal (LCD_CC) to make possible the control of the display's contrast by software. This is an 8-bit PWM (Pulse Width Modulation) signal that can be converted to an analog voltage with a simple passive filter.

The PWM module has a free-running counter whose value is compared against a compare register (CONSTRAST_VAL register). If the value in the counter is less than that in the register, the output brings the value of the polarity (POL) bit in the PWM control register: CONTRAST_CTR. Otherwise, the opposite value is output. Thus, a periodic waveform with a pulse width proportional to the value in the compare register is generated.

Due to the comparison mechanism, the output pulse has a width between zero and 255 PWM counter cycles. Thus by adding a simple passive filter outside the chip, an analog voltage between 0 and (255/256) × VDD can be obtained (for the positive polarity case, or between (1/256) × VDD and VDD for the negative polarity case). Other voltage values can be obtained by adding active external circuitry.

For PWM mode, the frequency of the counter can be adjusted to four different values using field PS of CONTRAST_CTR register.

37.5.3 LCD Interface

The LCD Controller interfaces with the LCD Module through the LCD Interface (Table 37-11 on page 580). The Controller supports the following interface configurations: 24-bit TFT single scan, 16-bit STN Dual Scan Mono (Color), 8-bit STN Dual (Single) Scan Mono (Color), 4-bit single scan Mono (Color).

A 4-bit single scan STN display uses 4 parallel data lines to shift data to successive single horizontal lines one at a time until the entire frame has been shifted and transferred. The 4 LSB pins of LCD Data Bus (LCDD [3:0]) can be directly connected to the LCD driver; the 20 MSB pins (LCDD [23:4]) are not used.

An 8-bit single scan STN display uses 8 parallel data lines to shift data to successive single horizontal lines one at a time until the entire frame has been shifted and transferred. The 8 LSB pins of LCD Data Bus (LCDD [7:0]) can be directly connected to the LCD driver; the 16 MSB pins (LCDD [23:8]) are not used.

An 8-bit Dual Scan STN display uses two sets of 4 parallel data lines to shift data to successive upper and lower panel horizontal lines one at a time until the entire frame has been shifted and transferred. The bus LCDD[3:0] is connected to the upper panel data lines and the bus LCDD[7:4] is connected to the lower panel data lines. The rest of the LCD Data Bus lines (LCDD[23:8]) are not used.

A 16-bit Dual Scan STN display uses two sets of 8 parallel data lines to shift data to successive upper and lower panel horizontal lines one at a time until the entire frame has been shifted and transferred. The bus LCDD[7:0] is connected to the upper panel data lines and the bus LCDD[15:8] is connected to the lower panel data lines. The rest of the LCD Data Bus lines (LCDD[23:16]) are not used.

STN Mono displays require one bit of image data per pixel. STN Color displays require three bits (Red, Green and Blue) of image data per pixel, resulting in a horizontal shift register of length three times the number of pixels per horizontal line. This RGB or Monochrome data is shifted to the LCD driver as consecutive bits via the parallel data lines.

A TFT single scan display uses up to 24 parallel data lines to shift data to successive horizontal lines one at a time until the entire frame has been shifted and transferred. The 24 data lines are divided in three bytes that define the color shade of each color component of each pixel. The LCDD bus is split as LCDD[23:16] for the blue component, LCDD[15:8] for the green component and LCDD[7:0] for the red component. If the LCD Module has lower color resolution (fewer bits per color component), only the most significant bits of each component are used.

All these interfaces are shown in Figure 37-8 to Figure 37-12. Figure 37-8 on page 576 shows the 24-bit single scan TFT display timing; Figure 37-9 on page 576 shows the 4-bit single scan STN display timing for monochrome and color modes; Figure 37-10 on page 577 shows the 8-bit single scan STN display timing for monochrome and color modes; Figure 37-11 on page 578 shows the 8-bit Dual Scan STN display timing for monochrome and color modes; Figure

37-12 on page 579 shows the 16-bit Dual Scan STN display timing for monochrome and color modes.

Figure 37-8. TFT Timing (First Line Expanded View)

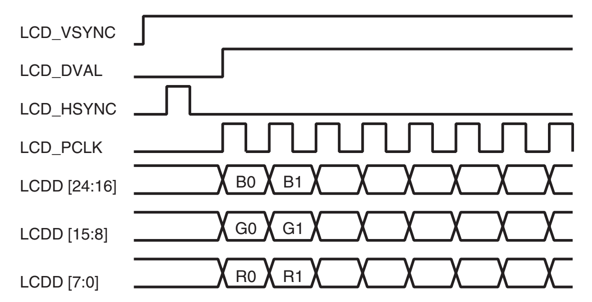


Figure 37-9. Single Scan Monochrome and Color 4-bit Panel Timing (First Line Expanded View)

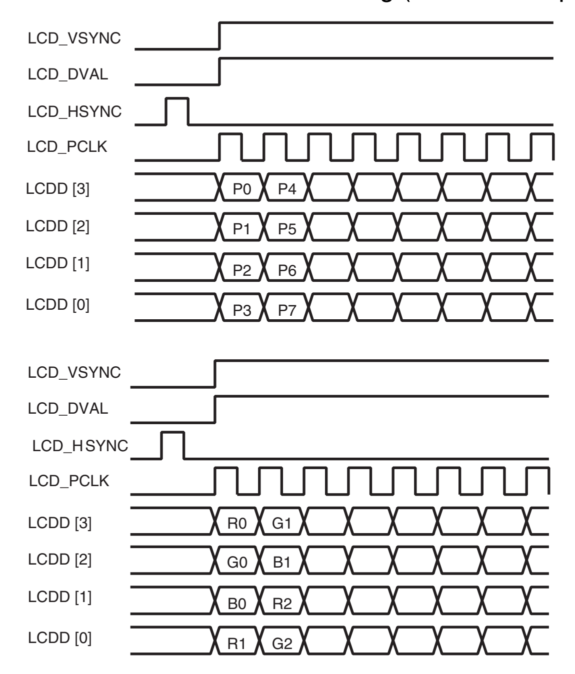


Figure 37-10. Single Scan Monochrome and Color 8-bit Panel Timing (First Line Expanded View)

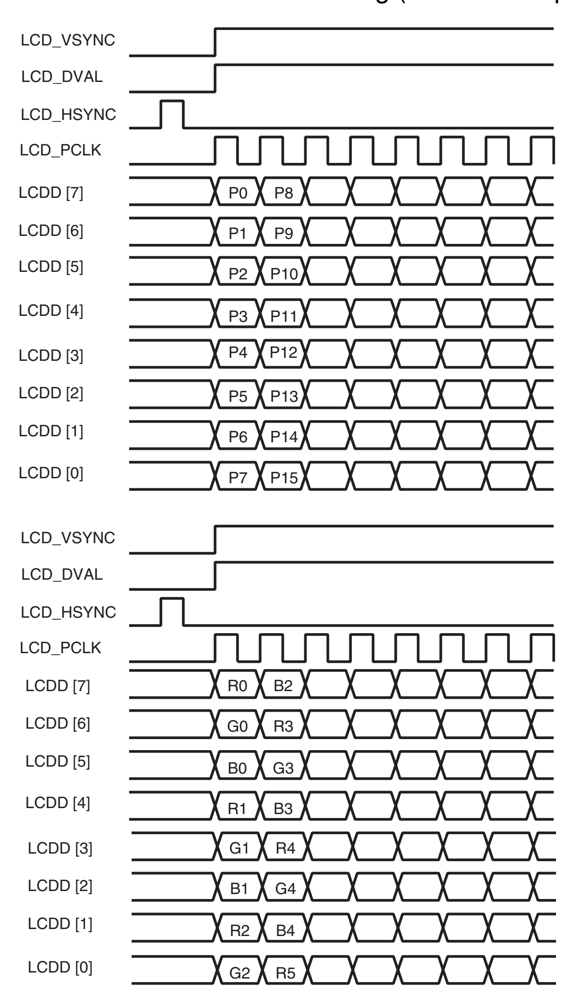


Figure 37-11. Dual Scan Monochrome and Color 8-bit Panel Timing (First Line Expanded View)

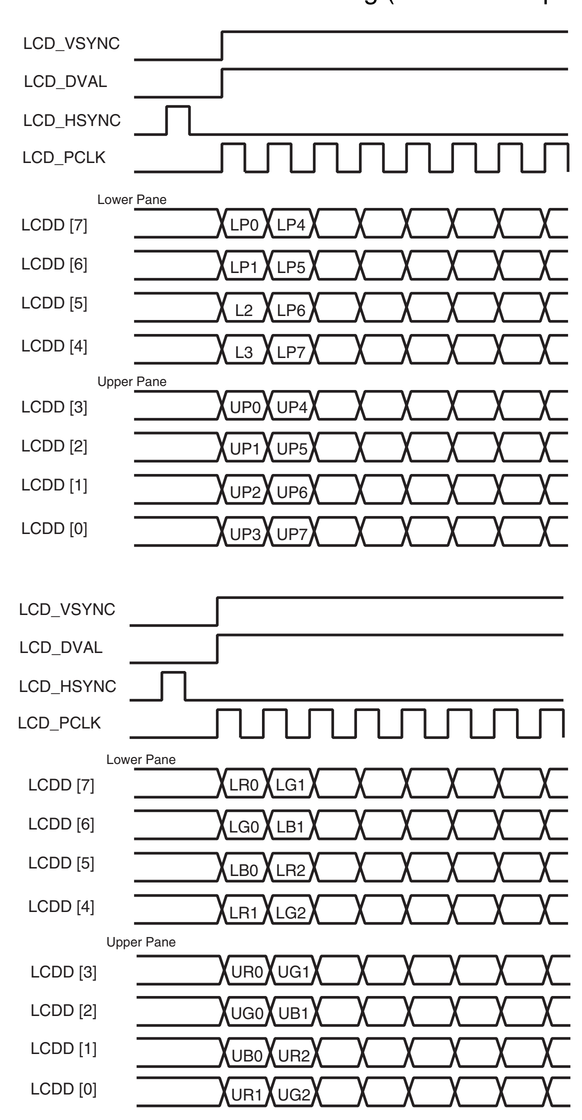


Figure 37-12. Dual Scan Monochrome and Color 16-bit Panel Timing (First Line Expanded View)

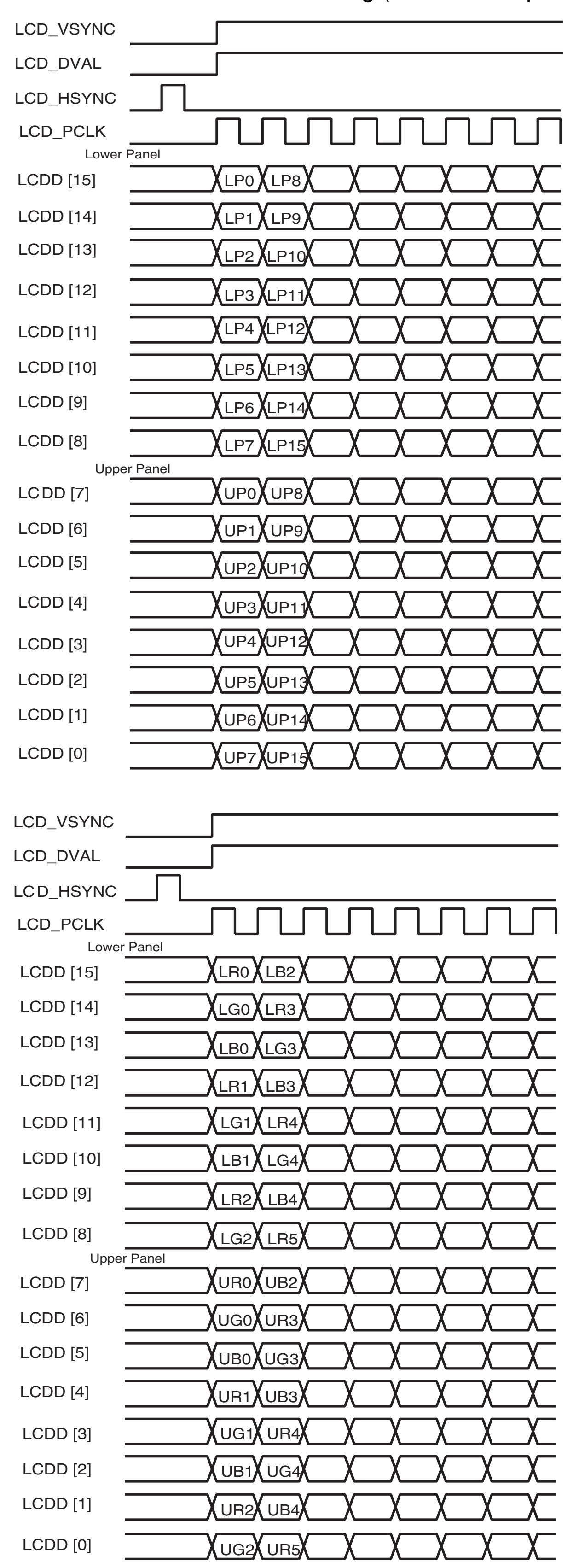




Table 37-11. LCD Signal Multiplexing

LCD Data Bus	4-bit STN Single Scan (mono, color)	8-bit STN Single Scan (mono, color)	8-bit STN Dual Scan (mono, color)	16-bit STN Dual Scan (mono, color)	24-bit TFT	16-bit TFT
LCDD[23]					LCD_BLUE7	LCD_BLUE4
LCDD[22]					LCD_BLUE6	LCD_BLUE3
LCDD[21]					LCD_BLUE5	LCD_BLUE2
LCDD[20]					LCD_BLUE4	LCD_BLUE1
LCDD[19]					LCD_BLUE3	LCD_BLUE0
LCDD[18]					LCD_BLUE2	Intensity Bit
LCDD[17]					LCD_BLUE1	
LCDD[16]					LCD_BLUE0	
LCDD[15]				LCDLP7	LCD_GREEN7	LCD_GREEN4
LCDD[14]				LCDLP6	LCD_GREEN6	LCD_GREEN3
LCDD[13]				LCDLP5	LCD_GREEN5	LCD_GREEN2
LCDD[12]				LCDLP4	LCD_GREEN4	LCD_GREEN1
LCDD[11]				LCDLP3	LCD_GREEN3	LCD_GREEN0
LCDD[10]				LCDLP2	LCD_GREEN2	Intensity Bit
LCDD[9]				LCDLP1	LCD_GREEN1	
LCDD[8]				LCDLP0	LCD_GREEN0	
LCDD[7]		LCD7	LCDLP3	LCDUP7	LCD_RED7	LCD_RED4
LCDD[6]		LCD6	LCDLP2	LCDUP6	LCD_RED6	LCD_RED3
LCDD[5]		LCD5	LCDLP1	LCDUP5	LCD_RED5	LCD_RED2
LCDD[4]		LCD4	LCDLP0	LCDUP4	LCD_RED4	LCD_RED1
LCDD[3]	LCD3	LCD3	LCDUP3	LCDUP3	LCD_RED3	LCD_RED0
LCDD[2]	LCD2	LCD2	LCDUP2	LCDUP2	LCD_RED2	Intensity Bit
LCDD[1]	LCD1	LCD1	LCDUP1	LCDUP1	LCD_RED1	
LCDD[0]	LCD0	LCD0	LCDUP0	LCDUP0	LCD_RED0	

37.6 Interrupts

The LCD Controller generates six different IRQs. All the IRQs are synchronized with the internal LCD Core Clock. The IRQs are:

- DMA Memory error IRQ. Generated when the DMA receives an error response from an AHB slave while it is doing a data transfer.
- FIFO underflow IRQ. Generated when the Serializer tries to read a word from the FIFO when the FIFO is empty.
- FIFO overwrite IRQ. Generated when the DMA Controller tries to write a word in the FIFO while the FIFO is full.
- DMA end of frame IRQ. Generated when the DMA controller updates the Frame Base Address pointers. This IRQ can be used to implement a double-buffer technique. For more information, see ”Double-buffer Technique” on page 583.
- End of Line IRQ. This IRQ is generated when the LINEBLANK period of each line is reached and the DMA Controller is in inactive state.
- End of Last Line IRQ. This IRQ is generated when the LINEBLANK period of the last line of the current frame is reached and the DMA Controller is in inactive state.

Each IRQ can be individually enabled, disabled or cleared, in the LCD_IER (Interrupt Enable Register), LCD_IDR (Interrupt Disable Register) and LCD_ICR (Interrupt Clear Register) registers. The LCD_IMR register contains the mask value for each IRQ source and the LDC_ISR contains the status of each IRQ source. A more detailed description of these registers can be found in ”LCD Controller (LCDC) User Interface” on page 585.

37.7 Configuration Sequence

The DMA Controller starts to transfer image data when the LCDC Core is activated (Write to LCD_PWR field of PWRCON register). Thus, the user should configure the LCDC Core and configure and enable the DMA Controller prior to activation of the LCD Controller. In addition, the image data to be shows should be available when the LCDC Core is activated, regardless of the value programmed in the GUARD_TIME field of the PWRCON register.

To disable the LCD Controller, the user should disable the LCDC Core and then disable the DMA Controller. The user should not enable LIP again until the LCDC Core is in IDLE state. This is checked by reading the LCD_BUSY bit in the PWRCON register.

The initialization sequence that the user should follow to make the LCDC work is:

- Create or copy the first image to show in the display buffer memory.
- If a palletized mode is used, create and store a palette in the internal LCD Palette memory(See “Palette” on page 566.
- Configure the LCD Controller Core without enabling it:
  - LCDCON1 register: Program the C*LKVAL* and *BYPASS* fields: these fields control the pixel clock divisor that is used to generate the pixel clock LCD_PCLK. The value to program depends on the LCD Core clock and on the type and size of the LCD Module used. There is a minimum value of the LCD_PCLK clock period that depends on the LCD Controller Configuration, this minimum value can be found in Table 37-10 on page 570. The equations that are used to calculate the value of the pixel clock divisor can be found at the end of the section ”Timegen” on page 569
  - LCDCON2 register: Program its fields following their descriptions in the LCD Controller User Interface section below and considering the type of LCD module



used and the desired working mode. Consider that not all combinations are possible.

– LCDTIM1 and LCDTIM2 registers: Program their fields according to the datasheet of the LCD module used and with the help of the Timegen section in page 10. Note that some fields are not applicable to STN modules and must be programmed with 0 values. Note also that there is a limitation on the minimum value of VHDLY, HPW, HBP that depends on the configuration of the LCDC.

– LCDFRMCFG register: program the dimensions of the LCD module used.

– LCDFIFO register: To program it, use the formula in section ”FIFO” on page 564

– LCDMVAL register: Its configuration depends on the LCD Module used and should be tuned to improve the image quality in the display (See “Timegen” on page 569.)

– DP1_2 to DP6_7 registers: they are only used for STN displays. They contain the dithering patterns used to generate gray shades or colors in these modules. They are loaded with recommended patterns at reset, so it is not necessary to write anything on them. They can be used to improve the image quality in the display by tuning the patterns in each application.

– PWRCON Register: this register controls the power-up sequence of the LCD, so take care to use it properly. Do not enable the LCD (writing a 1 in LCD_PWR field) until the previous steps and the configuration of the DMA have been finished.

– CONTRAST_CTR and CONTRAST_VAL: use this registers to adjust the contrast of the display, when the *lcd_cc* line is used.

- Configure the DMA Controller. The user should configure the base address of the display buffer memory, the size of the AHB transaction and the size of the display image in memory. When the DMA is configured the user should enable the DMA. To do so the user should configure the following registers:

  – DMABADDR1 and DMABADDR2 registers: In single scan mode only DMABADDR1 register must be configured with the base address of the display buffer in memory. In dual scan mode DMABADDR1 should be configured with the base address of the Upper Panel display buffer and DMABADDR2 should be configured with the base address of the Lower Panel display buffer.

  – DMAFRMCFG register: Program the FRMSIZE field. Note that in dual scan mode the vertical size to use in the calculation is that of each panel. Respect to the BRSTLN field, a recommended value is a 4-word burst.

  – DMACON register: Once both the LCD Controller Core and the DMA Controller have been configured, enable the DMA Controller by writing a “1” to the DMAEN field of this register.

- Finally, enable the LCD Controller Core by writing a “1” in the LCD_PWR field of the PWRCON register and do any other action that may be required to turn the LCD module on.

37.8 Double-buffer Technique

The double-buffer technique is used to avoid flickering while the frame being displayed is updated. Instead of using a single buffer, there are two different buffers, the backbuffer (background buffer) and the primary buffer (the buffer being displayed).

The host updates the backbuffer while the LCD Controller is displaying the primary buffer. When the backbuffer has been updated the host updates the DMA Base Address registers.

When using a Dual Panel LCD Module, both base address pointers should be updated in the same frame. There are two possibilities:

- Check the DMAFRMPTx register to ensure that there is enough time to update the DMA Base Address registers before the end of frame.
- Update the Frame Base Address Registers when the End Of Frame IRQ is generated.

Once the host has updated the Frame Base Address Registers and the next DMA end of frame IRQ arrives, the backbuffer and the primary buffer are swapped and the host can work with the new backbuffer.

37.9 Register Configuration Guide

37.9.1 STN Mode Example

STN color(R,G,B) 320*240, 8-bit single scan, 70 frames/sec, LCD_core clock = 60 Mhz

Data rate : 320*240*70*3/8 = 2.016MHz

HOZVAL= ((3*320)/8) -1

LINEVAL= 240 -1

CLKVAL = 60Mhz/ (2*2.016Mhz) = 15

LCDCON1= CLKVAL << 12

LCDCON2 = LITTLEENDIAN | SINGLESCAN | STNCOLOR | DISP8BIT| PS8BPP;

LCDTIM1 = 0;

LCDTIM2 = 10 | (10 << 21);

LCDFRMCFG = (HOZVAL << 21) | LINEVAL;

LCDMVAL = 0x80000004;

DMAFRMCFG = (7 << 24) + (320 * 240 * 8) / 32;

37.9.2 TFT Mode Example

This example is based on the NEC TFT color LCD module NL6448BC20-08.

TFT 640*480, 16-bit single scan, 60 frames/sec, pixel clock frequency = [21Mhz..29MHZ] with a typical value = 25,175Mhz.

The LCD_Core clock must be (2*n)*pixel clock frequency [42Mhz..100Mhz]

HOZVAL = 640 -1

LINEVAL = 240 -1

If LCD_Core clock is 50 Mhz

CLKVAL = 50Mhz/ (2*25,175Mhz) = 1

VFP = (12 -1), VBP = (31-1), VPW = (2-1), VHDLY= (2-1)

HFP = (16-1), HBP = (48 -1), HPW= (96-1)

LCDCON1= CLKVAL << 12

LCDCON2 = LITTLEENDIAN | CLKMOD | INVERT_CLK | INVERT_LINE | INVERT_FRM | PS16BPP | SINGLESCAN | TFT

LCDTIM1 = VFP | (VBP << 8) | (VPW << 16) | (VHDLY << 24)

LCDTIM2 = HBP | (HPW << 8) | (HFP << 21)

LCDFRMCFG = (HOZVAL << 21) | LINEVAL

LCDMVAL = 0

DMAFRMCFG = (7 << 24) + (640 * 480* 16) / 32;

37.10 LCD Controller (LCDC) User Interface

Table 37-12. LCD Controller (LCDC) User Interface

Offset	Register	Register Name	Access	Reset Value
0x0	DMA Base Address Register 1	DMABADDR1	R/W	0x00000000
0x4	DMA Base Address Register 2	DMABADDR2	R/W	0x00000000
0x8	DMA Frame Pointer Register 1	DMAFRMPT1	Read-only	0x00000000
0xC	DMA Frame Pointer Register 2	DMAFRMPT2	Read-only	0x00000000
0x10	DMA Frame Address Register 1	DMAFRMADD1	Read-only	0x00000000
0x14	DMA Frame Address Register 2	DMAFRMADD2	Read-only	0x00000000
0x18	DMA Frame Configuration Register	DMAFRMCFG	R/W	0x00000000
0x1C	DMA Control Register	DMACON	R/W	0x00000000
0x800	LCD Control Register 1	LCDCON1	R/W	0x00002000
0x804	LCD Control Register 2	LCDCON2	R/W	0x00000000
0x808	LCD Timing Register 1	LCDTIM1	R/W	0x00000000
0x80C	LCD Timing Register 2	LCDTIM2	R/W	0x00000000
0x810	LCD Frame Configuration Register	LCDFRMCFG	R/W	0x00000000
0x814	LCD FIFO Register	LCDFIFO	R/W	0x00000000
0x818	LCD_MODE Toggle Rate Value Register	LCDMVAL	R/W	0x00000000
0x81C	Dithering Pattern DP1_2	DP1_2	R/W	0xA5
0x820	Dithering Pattern DP4_7	DP4_7	R/W	0x5AF0FA5
0x824	Dithering Pattern DP3_5	DP3_5	R/W	0xA5A5F
0x828	Dithering Pattern DP2_3	DP2_3	R/W	0xA5F
0x82C	Dithering Pattern DP5_7	DP5_7	R/W	0xFAF5FA5
0x830	Dithering Pattern DP3_4	DP3_4	R/W	0xFAF5
0x834	Dithering Pattern DP4_5	DP4_5	R/W	0xFAF5F
0x838	Dithering Pattern DP6_7	DP6_7	R/W	0xF5FFAFF
0x83C	Power Control Register	PWRCON	R/W	0x0000000e
0x840	Contrast Control Register	CONTRAST_CTR	R/W	0x00000000
0x844	Contrast Value Register	CONTRAST_VAL	R/W	0x00000000
0x848	LCD Interrupt Enable Register	LCD_IER	Write-only	0x0
0x84C	LCD Interrupt Disable Register	LCD_IDR	Write-only	0x0
0x850	LCD Interrupt Mask Register	LCD_IMR	Read-only	0x0
0x854	LCD Interrupt Status Register	LCD_ISR	Read-only	0x0
0x858	LCD Interrupt Clear Register	LCD_ICR	Write-only	0x0
0xC00	Palette entry 0	LUT ENTRY 0	R/W	
0xC04	Palette entry 1	LUT ENTRY 1	R/W	
0xC08	Palette entry 2	LUT ENTRY 2	R/W	

Table 37-12. LCD Controller (LCDC) User Interface (Continued)

Offset	Register	Register Name	Access	Reset Value
0xC0C	Palette entry 3	LUT ENTRY 3	R/W	
…		…		
0xFFC	Palette entry 255	LUT ENTRY 255	R/W	

37.10.1 DMA Base Address Register 1

Name: DMABADDR1

Access: Read/Write

Reset value: 0x00000000

31	30	29	28	27	26	25	24
BADDR-U							
23	22	21	20	19	18	17	16
BADDR-U							
15	14	13	12	11	10	9	8
BADDR-U							
7	6	5	4	3	2	1	0
BADDR-U							

- **BADDR-U**

Base Address for the upper panel in dual scan mode. Base Address for the complete frame in single scan mode.

37.10.2 DMA Base Address Register 2

Name: DMABADDR2

Access: Read/Write

Reset value: 0x00000000

31	30	29	28	27	26	25	24
BADDR-L							
23	22	21	20	19	18	17	16
BADDR-L							
15	14	13	12	11	10	9	8
BADDR-L							
7	6	5	4	3	2	1	0
BADDR-L							

- **BADDR-L**

Base Address for the lower panel in dual scan mode only.

37.10.3 DMA Frame Pointer Register 1

Name: DMAFRMPT1

Access: Read-only

Reset value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	FRMPT-U						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FRMPT-U							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FRMPT-U							

- **FRMPT-U**

Current value of frame pointer for the upper panel in dual scan mode. Current value of frame pointer for the complete frame in single scan mode. Down count from FRMSIZE to 0.

Note: This register is read-only and contains the current value of the frame pointer (number of words to the end of the frame). It can be used as an estimation of the number of words transferred from memory for the current frame.

37.10.4 DMA Frame Pointer Register 2

Name: DMAFRMPT2

Access: Read-only

Reset value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	FRMPT-L						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FRMPT-L							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FRMPT-L							

- **FRMPT-L**

Current value of frame pointer for the Lower panel in dual scan mode only. Down count from FRMSIZE to 0.

Note: This register is read-only and contains the current value of the frame pointer (number of words to the end of the frame). It can be used as an estimation of the number of words transferred from memory for the current frame.

37.10.5 DMA Frame Address Register 1

Name: DMAFRMADD1

Access: Read-only

Reset value: 0x00000000

31	30	29	28	27	26	25	24
FRMADD-U							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
FRMADD-U							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FRMADD-U							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FRMADD-U							

- **FRMADD-U**

Current value of frame address for the upper panel in dual scan mode. Current value of frame address for the complete frame in single scan.

Note: This register is read-only and contains the current value of the last DMA transaction in the bus for the panel/frame.

37.10.6 DMA Frame Address Register 2

Name: DMAFRMADD2

Access: Read-only

Reset value: 0x00000000

31	30	29	28	27	26	25	24
FRMADD-L							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
FRMADD-L							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FRMADD-L							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FRMADD-L							

- **FRMADD-L**

Current value of frame address for the lower panel in single scan mode only.

Note: This register is read-only and contains the current value of the last DMA transaction in the bus for the panel.

37.10.7 DMA Frame Configuration Register

Name: DMAFRMCFG

Access: Read/Write

Reset value: 0x00000000

31	30	29	28	27	26	25	24
–	BRSTLN						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
-	FRMSIZE						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FRMSIZE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FRMSIZE							

- **FRMSIZE: Frame Size**

In single scan mode, this is the frame size in words. In dual scan mode, this is the size of each panel.

- **BRSTLN: Burst Length**

Program with the desired burst length - 1

37.10.8 DMA Control Register

Name: DMACON

Access: Read/Write

Reset value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	DMABUSY	DMARST	DMAEN

- **DMAEN: DMA Enable**

0: DMA is disabled.

1: DMA is enabled.

- **DMARST: DMA Reset (Write-only)**

0: No effect.

1: Reset DMA module. DMA Module should be reset only when disabled and in idle state.

- **DMABUSY: DMA Busy**

0: DMA module is idle.

1: DMA module is busy (doing a transaction on the AHB bus).

37.10.9 LCD Control Register 1

Name: LCDCON1

Access: Read/Write, except LINECNT: Read-only

Reset value: 0x00002000

31	30	29	28	27	26	25	24
LINECNT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
LINECNT			CLKVAL				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CLKVAL				–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	BYPASS

- **BYPASS: Bypass lcd_pclk divider**

0: The divider is not bypassed. LCD_PCLK frequency defined by the CLKVAL field.

1: The LCD_PCLK divider is bypassed. LCD_PCLK frequency is equal to the LCDC Clock frequency.

- **CLKVAL: Clock divider**

9-bit divider for pixel clock (lcd_pclk) frequency.

$$Pixel_clock = system_clock / (CLKVAL + 1) \times 2$$

- **LINECNT: Line Counter (Read-only)**

Current Value of 11-bit line counter. Down count from LINEVAL to 0.

37.10.10 LCD Control Register 2

Name: LCDCON2

Access: Read/Write

Reset value: 0x0000000

31	30	29	28	27	26	25	24
MEMOR		–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CLKMOD	–	–	INVDVAL	INVCLK	INVLINE	INVFRAME	INVVD
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PIXELSIZE			IFWIDTH		SCANMOD	DISTYPE	

- **DISTYPE: Display Type**

DISTYPE		
0	0	STN Monochrome
0	1	STN Color
1	0	TFT
1	1	Reserved

- **SCANMOD: Scan Mode**

0: Single Scan

1: Dual Scan

- **IFWIDTH: Interface width (STN)**

IFWIDTH		
0	0	4-bit (Only valid in single scan STN mono or color)
0	1	8-bit (Only valid in STN mono or Color)
1	0	16-bit (Only valid in dual scan STN mono or color)
1	1	Reserved

- **PIXELSIZE: Bits per pixel**

PIXELSIZE			
0	0	0	1 bit per pixel
0	0	1	2 bits per pixel
0	1	0	4 bits per pixel
0	1	1	8 bits per pixel
1	0	0	16 bits per pixel
1	0	1	24 bits per pixel (Only valid in TFT mode)
1	1	0	Reserved
1	1	1	Reserved

- **INVVD: LCDD polarity**

0: Normal

1: Inverted

- **INVFRAME: lcd_vsync polarity**

0: Normal (active high)

1: Inverted (active low)

- **INVLINE: lcd_hsync polarity**

0: Normal (active high)

1: Inverted (active low)

- **INVCLK: lcd_pclk polarity**

0: Normal (LCDD fetched at lcd_pclk falling edge)

1: Inverted (LCDD fetched at lcd_pclk rising edge)

- **INVDVAL: lcd_dval polarity**

0: Normal (active high)

1: Inverted (active low)

- **CLKMOD: lcd_pclk mode**

0: lcd_pclk only active during active display period

1: lcd_pclk always active

- **MEMOR: Memory Ordering Format**

00: Big Endian

10: Little Endian

37.10.11 LCD Timing Configuration Register 1

Name: LCDTIM1

Access: Read/Write

Reset value: 0x0000000

31	30	29	28	27	26	25	24
–	–	–	–	VHDLY			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	VPW					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
VBP							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VFP							

- **VFP: Vertical Front Porch**

In TFT mode, these bits equal the number of idle lines at the end of the frame.

In STN mode, these bits should be set to 0.

- **VBP: Vertical Back Porch**

In TFT mode, these bits equal the number of idle lines at the beginning of the frame.

In STN mode, these bits should be set to 0.

- **VPW: Vertical Synchronization pulse width**

In TFT mode, these bits equal the vertical synchronization pulse width, given in number of lines. LCD_VSYNC width is equal to (VPW+1) lines.

In STN mode, these bits should be set to 0.

- **VHDLY: Vertical to horizontal delay**

In TFT mode, this is the delay between LCD_VSYNC rising or falling edge and LCD_HSYNC rising edge. Delay is (VHDLY+1) LCD_PCLK cycles.

In STN mode, these bits should be set to 0.



37.10.12 LCD Timing Configuration Register 2

Name: LCDTIM2

Access: Read/Write

Reset value: 0x0000000

31	30	29	28	27	26	25	24
HFP							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
HFP			–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	HPW					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HBP							

- **HBP: Horizontal Back Porch**

Number of idle LCD_PCLK cycles at the beginning of the line. Idle period is (HBP+1) LCD_PCLK cycles.

- **HPW: Horizontal synchronization pulse width**

Width of the LCD_HSYNC pulse, given in LCD_PCLK cycles. Width is (HPW+1) LCD_PCLK cycles.

- **HFP: Horizontal Front Porch**

Number of idle lcd_pclk cycles at the end of the line. Idle period is (HFP+1) lcd_pclk cycles.

37.10.13 LCD Frame Configuration Register

Name: LCDFRMCFG

Access: Read/Write

Reset value: 0x0000000

31	30	29	28	27	26	25	24
HOZVAL							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
HOZVAL			–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	LINEVAL		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LINEVAL							

- **LINEVAL: Vertical size of LCD module**

LINEVAL = (Vertical display size) - 1

In dual scan mode, vertical display size refers to the size of each panel.

- **HOZVAL: Horizontal size of LCD module**

In STN Mode:

– HOZVAL = (Horizontal display size / Number of valid LCDD data line) - 1

– In STN monochrome mode, Horizontal display size = Number of horizontal pixels

– In STN color mode, Horizontal display size = 3*Number of horizontal pixels

– In 4-bit single scan or 8-bit dual scan STN display mode, number of valid LCDD data lines = 4

– In 8-bit single scan or 16-bit dual scan STN display mode, number of valid LCDD data lines = 8

– If the value calculated for HOZVAL with the above formula is not an integer, it must be rounded up to the next integer value.

In TFT mode:

– HOZVAL = Horizontal display size



37.10.14 LCD FIFO Register

Name: LCDFIFO

Access: Read/Write

Reset value: 0x0000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FIFOTH							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FIFOTH							

- **FIFOTH: FIFO Threshold**

Must be programmed with:

FIFOTH = 512 words - (2 x DMA_burst_length + 3)

where:

- 512 words is the effective size of the FIFO. It is the total FIFO memory size in single scan mode and half that size in dual scan mode.
- DMA_burst_length is the burst length of the transfers made by the DMA.

37.10.15 LCD_MODE Toggle Rate Value Register

Name: LCDMVAL

Access: Read/Write

Reset value: 0x00000000

31	30	29	28	27	26	25	24
MMODE	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
MVAL							

- **MVAL: LCD_MODE toggle rate value**

LCD_MODE toggle rate if MMODE = 1. Toggle rate is MVAL + 1 line periods.

- **MMODE: LCD_MODE toggle rate select**

0: Each Frame

1: Rate defined by MVAL

37.10.16 Dithering Pattern DP1_2 Register

Name: DP1_2

Access: Read/Write

Reset value: 0xA5

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DP1_2							

• **DP1_2: Pattern value for ½ duty cycle**

37.10.17 Dithering Pattern DP4_7 Register

Name: DP4_7

Access: Read/Write

Reset value: 0x5AF0FA5

31	30	29	28	27	26	25	24
–	–	–	–	DP4_7			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DP4_7							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DP4_7							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DP4_7							

• **DP4_7: Pattern value for 4/7 duty cycle**

37.10.18 Dithering Pattern DP3_5 Register

Name: DP3_5

Access: Read/Write

Reset value: 0xA5A5F

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	DP3_5			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DP3_5							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DP3_5							

• **DP3_5: Pattern value for 3/5 duty cycle**

37.10.19 Dithering Pattern DP2_3 Register

Name: DP2_3: Dithering Pattern DP2_3 Register

Access: Read/Write

Reset value: 0xA5F

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	DP2_3			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DP2_3							

- **DP2_3: Pattern value for 2/3 duty cycle**

37.10.20 Dithering Pattern DP5_7 Register

Name: DP5_7:

Access: Read/Write

Reset value: 0xFAF5FA5

31	30	29	28	27	26	25	24
–	–	–	–	DP5_7			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DP5_7							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DP5_7							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DP5_7							

- **DP5_7: Pattern value for 5/7 duty cycle**

37.10.21 Dithering Pattern DP3_4 Register

Name: DP3_4

Access: Read/Write

Reset value: 0xFAF5

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DP3_4							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DP3_4							

- **DP3_4: Pattern value for 3/4 duty cycle**



37.10.22 Dithering Pattern DP4_5 Register

Name: DP4_5

Access: Read/Write

Reset value: 0xFAF5F

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	DP4_5			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DP4_5							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DP4_5							

- **DP4_5: Pattern value for 4/5 duty cycle**

37.10.23 Dithering Pattern DP6_7 Register

Name: DP6_7

Access: Read/Write

Reset value: 0xF5FFAFF

31	30	29	28	27	26	25	24
–	–	–	–	DP6_7			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DP6_7							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DP6_7							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DP6_7							

- **DP6_7: Pattern value for 6/7 duty cycle**

37.10.24 Power Control Register

Name: PWRCON

Access: Read/Write

Reset value: 0x0000000e

31	30	29	28	27	26	25	24
LCD_BUSY	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
GUARD_TIME							LCD_PWR

- **LCD_PWR: LCD module power control**

0 = lcd_pwr pin is low, other LCD_* pins are low.

0->1 = lcd_* pins activated, lcd_pwr are set high with the delay of GUARD_TIME frame periods.

1 = lcd_pwr pin is high, other lcd_* pins are active

1->0 = lcd_pwr pin is low, other lcd_* pins are active, but are set low after GUARD_TIME frame periods.

- **GUARD_TIME**

Delay in frame periods between applying control signals to the LCD module and setting LCD_PWR high, and between setting LCD_PWR low and removing control signals from LCD module

- **LCD_BUSY**

Read-only field. If 1, it indicates that the LCD is busy (active and displaying data, in power_on sequence or in power off sequence).

37.10.25 Contrast Control Register

Name: CONTRAST_CTR

Access: Read/Write

Reset value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	ENA	POL	PS	

- **PS**

This 2-bit value selects the configuration of a counter prescaler. The meaning of each combination is as follows:

PS		
0	0	The counter advances at a rate of fCOUNTER = fLCDC_CLOCK.
0	1	The counter advances at a rate of fCOUNTER = fLCDC_CLOC /2.
1	0	The counter advances at a rate of fCOUNTER = fLCDC_CLOCK/4.
1	1	The counter advances at a rate of fCOUNTER = fLCDC_CLOCK/8.

- **POL**

This bit defines the polarity of the output. If 1, the output pulses are high level (the output will be high whenever the value in the counter is less than the value in the compare register CONSTRAST_VAL). If 0, the output pulses are low level.

- **ENA**

When 1, this bit enables the operation of the PWM generator. When 0, the PWM counter is stopped.

37.10.26 Contrast Value Register

Name: CONSTRAST_VAL

Access: Read/Write

Reset value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CVAL							

- **CVAL**

PWM compare value. Used to adjust the analog value obtained after an external filter to control the contrast of the display.

37.10.27 LCD Interrupt Enable Register

Name: LCD_IER

Access: Write-only

Reset value: 0x0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	MERIE	OWRIE	UFLWIE	-	EOFIE	LSTLNIE	LNIE

- **LNIE: Line interrupt enable**

0: No effect

1: Enable each line interrupt

- **LSTLNIE: Last line interrupt enable**

0: No effect

1: Enable last line interrupt

- **EOFIE: DMA End of frame interrupt enable**

0: No effect

1: Enable End Of Frame interrupt

- **UFLWIE: FIFO underflow interrupt enable**

0: No effect

1: Enable FIFO underflow interrupt

- **OWRIE: FIFO overwrite interrupt enable**

0: No effect

1: Enable FIFO overwrite interrupt

- **MERIE: DMA memory error interrupt enable**

0: No effect

1: Enable DMA memory error interrupt

37.10.28 LCD Interrupt Disable Register

Name: LCD_IDR

Access: Write-only

Reset value: 0x0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	MERID	OWRID	UFLWID	–	EOFID	LSTLNID	LNID

- **LNID: Line interrupt disable**

0: No effect

1: Disable each line interrupt

- **LSTLNID: Last line interrupt disable**

0: No effect

1: Disable last line interrupt

- **EOFID: DMA End of frame interrupt disable**

0: No effect

1: Disable End Of Frame interrupt

- **UFLWID: FIFO underflow interrupt disable**

0: No effect

1: Disable FIFO underflow interrupt

- **OWRID: FIFO overwrite interrupt disable**

0: No effect

1: Disable FIFO overwrite interrupt

- **MERID: DMA Memory error interrupt disable**

0: No effect

1: Disable DMA Memory error interrupt

37.10.29 LCD Interrupt Mask Register

Name: LCD_IMR

Access: Read-only

Reset value: 0x0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	MERIM	OWRIM	UFLWIM	–	EOFIM	LSTLNIM	LNIM

- **LNIM: Line interrupt mask**

0: Line Interrupt disabled

1: Line interrupt enabled

- **LSTLNIM: Last line interrupt mask**

0: Last Line Interrupt disabled

1: Last Line Interrupt enabled

- **EOFIM: DMA End of frame interrupt mask**

0: End Of Frame interrupt disabled

1: End Of Frame interrupt enabled

- **UFLWIM: FIFO underflow interrupt mask**

0: FIFO underflow interrupt disabled

1: FIFO underflow interrupt enabled

- **OWRIM: FIFO overwrite interrupt mask**

0: FIFO overwrite interrupt disabled

1: FIFO overwrite interrupt enabled

- **MERIM: DMA Memory error interrupt mask**

0: DMA Memory error interrupt disabled

1: DMA Memory error interrupt enabled

37.10.30 LCD Interrupt Status Register

Name: LCD_ISR

Access: Read-only

Reset value: 0x0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	MERIS	OWRIS	UFLWIS	–	EOFIS	LSTLNIS	LNIS

- **LNIS: Line interrupt status**

0: Line Interrupt not active

1: Line Interrupt active

- **LSTLNIS: Last line interrupt status**

0: Last Line Interrupt not active

1: Last Line Interrupt active

- **EOFIS: DMA End of frame interrupt status**

0: End Of Frame interrupt not active

1: End Of Frame interrupt active

- **UFLWIS: FIFO underflow interrupt status**

0: FIFO underflow interrupt not active

1: FIFO underflow interrupt active

- **OWRIS: FIFO overwrite interrupt status**

0: FIFO overwrite interrupt not active

1: FIFO overwrite interrupt active

- **MERIS: DMA Memory error interrupt status**

0: DMA Memory error interrupt not active

1: DMA Memory error interrupt active

37.10.31 LCD Interrupt Clear Register

Name: LCD_ICR

Access: Write-only

Reset value: 0x0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	MERIC	OWRIC	UFLWIC	–	EOFIC	LSTLNIC	LNIC

- **LNIC: Line interrupt clear**

0: No effect

1: Clear each line interrupt

- **LSTLNIC: Last line interrupt clear**

0: No effect

1: Clear Last line Interrupt

- **EOFIC: DMA End of frame interrupt clear**

0: No effect

1: Clear End Of Frame interrupt

- **UFLWIC: FIFO underflow interrupt clear**

0: No effect

1: Clear FIFO underflow interrupt

- **OWRIC: FIFO overwrite interrupt clear**

0: No effect

1: Clear FIFO overwrite interrupt

- **MERIC: DMA Memory error interrupt clear**

0: No effect

1: Clear DMA Memory error interrupt

38. AT91SAM9261 Electrical Characteristics

38.1 Absolute Maximum Ratings

Table 38-1. Absolute Maximum Ratings*

Operating Temperature (Industrial)	-40° C to +85° C
Storage Temperature	-60°C to +150°C
Voltage on Input Pins with Respect to Ground	-0.3V to +5.5V
Maximum Operating Voltage (VDDCORE and VDDBU)	1.32V
Maximum Operating Voltage (VDDOSC, VDDPLL, VDDIOM and VDDIOP)	3.6V
Total DC Output Current on all I/O lines	350mA

*NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

38.2 DC Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C, unless otherwise specified and are certified for a junction temperature up to T_J = 100°C.

Table 38-2. DC Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{VDDCORE}$	DC Supply Core		1.08		1.32	V
V_{VDDBU}	DC Supply Backup		1.08		1.32	
V_{VDDOSC}	DC Supply Oscillator		2.7		3.6	V
V_{VDDPLL}	DC Supply PLL		2.7		3.6	V
V_{VDDIOM}	DC Supply Memory I/Os		2.7		3.6	V
V_{VDDIOP}	DC Supply Peripheral I/Os		2.7		3.6	V
V_{IL}	Input Low-level Voltage		-0.3		0.8	V
V_{IH}	Input High-level Voltage	V_{VDDIO}= V_{VDDIOM} or V_{VDDIOP}	2		V_{VDDIO}+0.3	V
V_{OL}	Output Low-level Voltage				0.4	V
V_{OH}	Output High-level Voltage	V_{VDDIO}= V_{VDDIOM} or V_{VDDIOP}	V_{VDDIO}-0.4			V
I_{LEAK}	Input Leakage Current	Pullup resistors disabled			±1	µA
C_{IN}	Input Capacitance	217-ball LFBGA Package			5.0	pF
R_{PULLUP}	Pull-up Resistance	PA0-PA31, PB0-PB31, PC0-PC31	70	10	175	kOhm
I_O	Output Current	PA0-PA31, PB0-PB31, PC0-PC31			8	mA



Table 38-2. DC Characteristics (Continued)

I_{SC}	Static Current	On $V_{VDDCORE}$ = 1.2V, MCK = 0 Hz, excluding POR All inputs driven TMS, TDI, TCK, NRST = 1	T_A =25°C		345		µA
			T_A =85°C			3000	µA
		On V_{VDDBU} = 1.2V, Logic cells consumption, excluding POR All inputs driven WKUP = 0	T_A =25°C		1.13		µA
			T_A =85°C			9.8	µA

38.3 Power Consumption

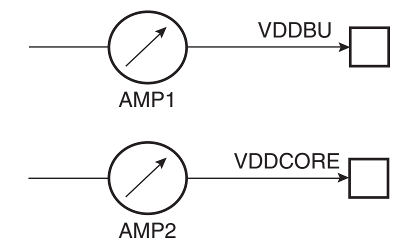
- Typical power consumption of PLLs, Slow Clock and Main Oscillator.
- Power consumption of power supply in four different modes: Active, Idle, Ultra Low-power and Backup.
- Power consumption by peripheral: calculated as the difference in current measurement after having enabled then disabled the corresponding clock.

38.3.1 Power Consumption versus Modes

The values in Table 38-3 and Table 38-4 on page 613 are measured values of power consumption with operating conditions as follows:

- $V_{DDIOM} = V_{DDIOP}$ = 3.3V
- $V_{DDPLL} = V_{DDOSC}$ = 3.3V
- $V_{DDCORE} = V_{DDBU}$ = 1.2V
- T_A = 25° C
- There is no consumption on the I/Os of the device

Figure 38-1. Measures Schematics



These figures represent the power consumption measured on the power supplies.

Table 38-3. Power Consumption for Different Modes(1)

Mode	Conditions	Consumption	Unit
Active	ARM Core clock is 200 MHz. MCK is 100 MHz. All peripheral clocks activated, including HCLK0. Cache de-activated. onto AMP2	64.1	mA
Idle	ARM core in idle state, waiting an interrupt. All peripheral clocks de-activated, including HCLK0. Cache de-activated. onto AMP2	11.2	mA
Ultra low power	ARM Core clock is 500 Hz. All peripheral clocks de-activated, including HCLK0. PLLA and PLLB de-activated. onto AMP2	400	µA
Backup	Device only V_{DDBU} powered onto AMP1	2.5	µA

Table 38-4. Power Consumption by Peripheral in Active Mode

Peripheral	Consumption	Unit
PIO Controller	0.5	mA
USART	1.5	
UHP	2.8	
UDP	1.3	
LCDC	1.3	
TWI	0.3	
SPI	0.9	
MCI	1.3	
SSC	1.6	
Timer Counter Channels	0.3	

38.4 Clock Characteristics

These parameters are given in the following conditions:

- V_{DDCORE} = 1.2V
- Ambient Temperature = 25°C

The Temperature Derating Factor described in ”Applicable Conditions and Derating Data” on page 618, section ”Temperature Derating Factor” on page 619 and V_{DDCORE} Voltage Derating Factor described in ”Applicable Conditions and Derating Data” on page 618, section ”VDDCORE Voltage Derating Factor” on page 619 are both applicable to these characteristics.

38.4.1 Processor Clock Characteristics

Table 38-5. Processor Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPPCK})$	Processor Clock Frequency			238	MHz
t_{CPPCK}	Processor Clock Period		4.19		ns

38.4.2 Master Clock Characteristics

Table 38-6. Master Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPMCK})$	Master Clock Frequency			119	MHz
t_{CPMCK}	Master Clock Period		8.38		ns

38.4.3 XIN Clock Characteristics

Table 38-7. XIN Clock Electrical Characteristics

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPXIN})$	XIN Clock Frequency			50.0	MHz
t_{CPXIN}	XIN Clock Period		20.0		ns
t_{CHXIN}	XIN Clock High Half-period		0.4 x t_{CPXIN}	0.6 x t_{CPXIN}	
t_{CLXIN}	XIN Clock Low Half-period		0.4 x t_{CPXIN}	0.6 x t_{CPXIN}	
C_{IN}	XIN Input Capacitance	(1)		25	pF
R_{IN}	XIN Pulldown Resistor	(1)		500	kΩ

Note: 1. These characteristics apply only when the Main Oscillator is in bypass mode (i.e., when MOSCEN = 0 and OSCBYPASS = 1 in the CKGR_MOR register.)

38.5 Crystal Oscillator Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C and worst case of power supply, unless otherwise specified.

38.5.1 32 kHz Oscillator Characteristics

Table 38-8. 32 kHz Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CP32KHz})$	Crystal Oscillator Frequency			32 768		KHz
	Duty Cycle		40		60	%
t_{ST}	Startup Time	V_{DDOSC} = 3.3V R_S = 50 kΩ, C_L = 12.5 pF(1)			900	ms

Note: 1. R_S is the equivalent series resistance, C_L is the equivalent load capacitance.

38.5.2 Main Oscillators Characteristics

Table 38-9. Main Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CPMAIN})$	Crystal Oscillator Frequency		3	16	20	MHz
C_{L1}, C_{L2}	Internal Load Capacitance ($C_{L1} = C_{L2}$)			25		pF
C_L	Equivalent Load Capacitance			12.5		pF
	Duty Cycle		40	50	60	%
t_{ST}	Startup Time	V_{DDPLL} = 3 to 3.6V C_S = 3 pF(1) $1/(t_{CPMAIN})$ = 3 MHz C_S = 7 pF(1) $1/(t_{CPMAIN})$ = 8MHz C_S = 7 pF(1) $1/(t_{CPMAIN})$ = 16 MHz C_S = 7 pF(1) $1/(t_{CPMAIN})$ = 20 MHz			 14.5 4 1.4 1	ms
I_{OSC}	Current Consumption	Active mode @20MHz		450	650	µA
		Standby mode @3.6V			1	µA

Notes: 1. C_S is the shunt capacitance

38.5.3 PLL Characteristics

Table 38-10. Phase Lock Loop Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
F_{OUT}	Output Frequency	Field out of CKGR_PLL is 00	80		200	MHz
		Field out of CKGR_PLL is 10	190		240	MHz
F_{IN}	Input Frequency		1		32	MHz
I_{PLL}	Current Consumption	active mode			3	mA
		standby mode			1	µA

Note: 1. Startup time depends on PLL RC filter. A calculation tool is provided by Atmel.



38.6 USB Transceiver Characteristics

38.6.1 Electrical Characteristics

Table 38-11. Electrical Parameters

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Input Levels						
V_{IL}	Low Level				0.8	V
V_{IH}	High Level		2.0			V
V_{DI}	Differential Input Sensivity	|(D+) - (D-)|	0.2			V
V_{CM}	Differential Input Common Mode Range		0.8		2.5	V
C_{IN}	Transceiver capacitance	Capacitance to ground on each line			9.18	pF
I	Hi-Z State Data Line Leakage	$0V < V_{IN} < 3.3V$	- 10		+ 10	µA
R_{EXT}	Recommended External USB Series Resistor	In series with each USB pin with ±5%		27		Ω
Output Levels						
V_{OL}	Low Level Output	Measured with R_L of 1.425 kΩ tied to 3.6V	0.0		0.3	V
V_{OH}	High Level Output	Measured with R_L of 14.25 kΩ tied to GND	2.8		3.6	V
V_{CRS}	Output Signal Crossover Voltage	Measure conditions described in Figure 38-2	1.3		2.0	V

38.6.2 Switching Characteristics

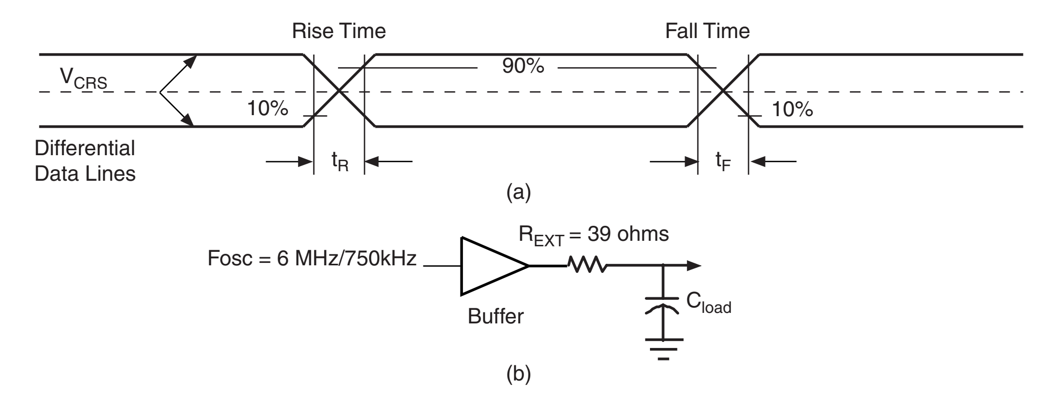
Table 38-12. In Low Speed

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{FR}	Transition Rise Time	C_{LOAD} = 400 pF	75		300	ns
t_{FE}	Transition Fall Time	C_{LOAD} = 400 pF	75		300	ns
t_{FRFM}	Rise/Fall time Matching	C_{LOAD} = 400 pF	80		125	%

Table 38-13. In Full Speed

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{FR}	Transition Rise Time	C_{LOAD} = 50 pF	4		20	ns
t_{FE}	Transition Fall Time	C_{LOAD} = 50 pF	4		20	ns
t_{FRFM}	Rise/Fall time Matching		90		111.11	%

Figure 38-2. USB Data Signal Rise and Fall Times



39. AT91SAM9261 AC Characteristics

39.1 Applicable Conditions and Derating Data

These conditions and derating process apply to the following paragraphs: Clock Characteristics and JTAG/ICE Timings.

39.1.1 Conditions and Timings Computation

The delays are given as typical values under the following conditions:

- V_{DDIOM} = 3.3V
- V_{DDCORE} = 1.2V
- Ambient Temperature = 25°C
- Load Capacitance = 0 pF
- The output level change detection is (0.5 x V_{DDIOM}).
- The input level is 0.8V for a low-level detection and is 2.0V for a high-level detection.

The minimum and maximum values given in the AC characteristics tables of this datasheet take into account process variation and design. In order to obtain the timingfor other conditions, the following equation should be used:

$$t = \delta_{T^\circ} \times \left((\delta_{VDDCORE} \times t_{DATASHEET}) + \left(\delta_{VDDIOM} \times \sum C_{SIGNAL} \times \delta_{CSIGNAL} \right) \right)$$

where:

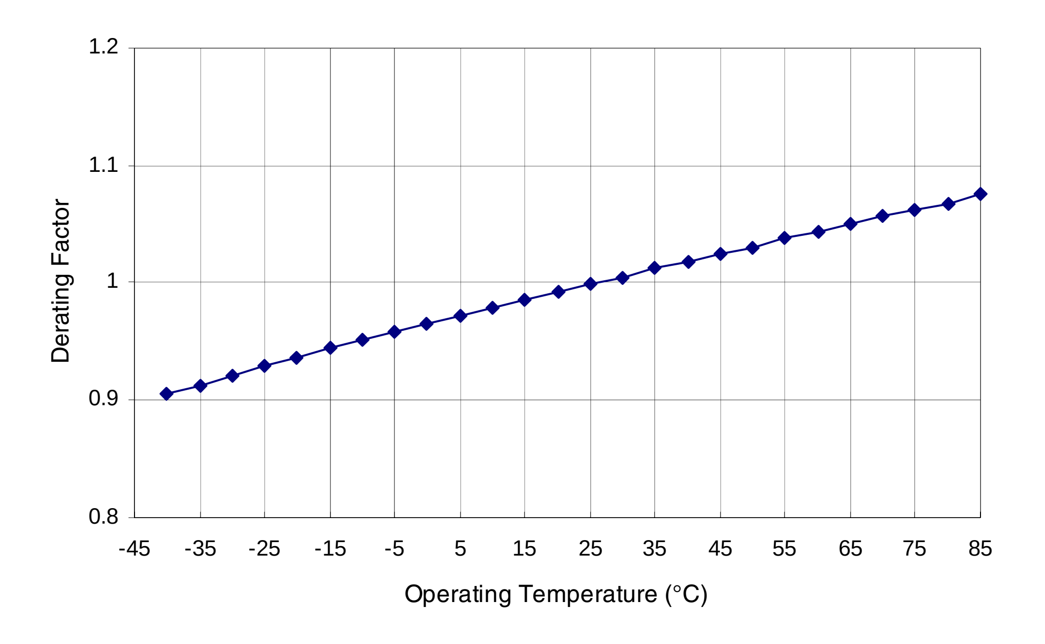
- δ_{T° is the derating factor in temperature given in Figure 39-1 on page 619.
- $\delta_{VDDCORE}$ is the derating factor for the Core Power Supply given in Figure 39-2 on page 619.
- $t_{DATASHEET}$ is the minimum or maximum timing value given in this datasheet for a load capacitance of 0 pF.
- δ_{VDDIOM} is the derating factor for the IOM Power Supply given in Figure 39-3 on page 620.
- C_{SIGNAL} is the capacitance load on the considered output pin[1].
- $\delta_{CSIGNAL}$ is the load derating factor depending on the capacitance load on the related output pins given in Min and Max in this datasheet.

The input delays are given as typical values.

Note: 1. The user must take into account the package capacitance load contribution (C_{IN}) described in "DC Characteristics" on page 611, Table 38-2 on page 611.

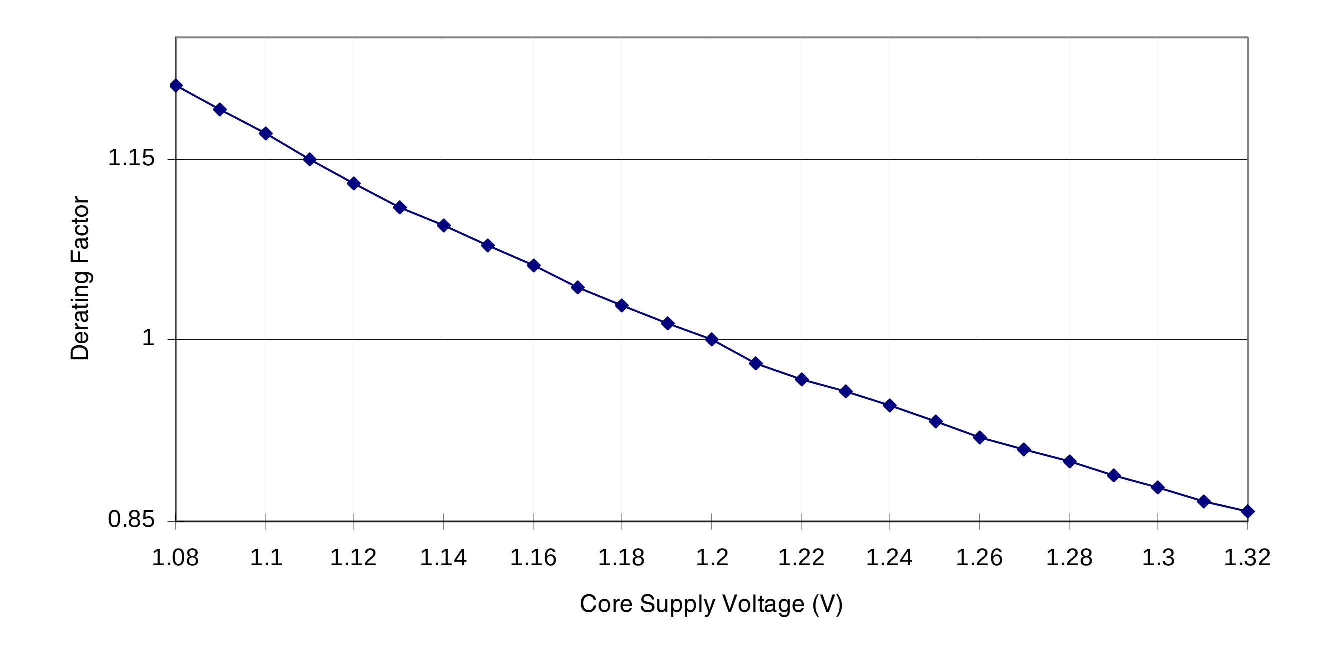
39.1.2 Temperature Derating Factor

Figure 39-1. Derating Curve for Different Operating Temperatures



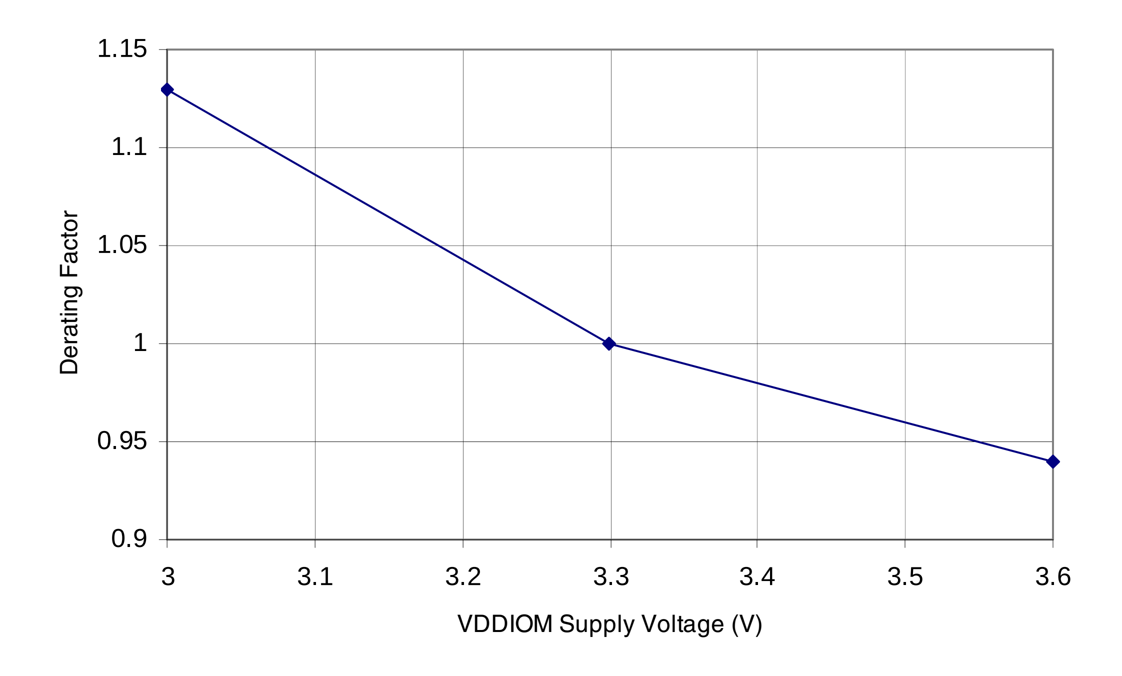
39.1.3 V_{DDCORE} Voltage Derating Factor

Figure 39-2. Derating Curve for Different Core Supply Voltages



39.1.4 V_{DDIOM} and V_{DDIOP} Voltage Derating Factor

Figure 39-3. Derating Curve for Different IO Supply Voltages)



Note: The derating factor in this example is applicable only to timings related to output pins.

39.2 EBI Timings

39.2.1 SMC Signals Relative to MCK

The following tables show timings relative to operating condition limits defined in the section "Conditions and Timings Computation" on page 618.

Table 39-1. Access Type Independent SMC Signals

Symbol	Parameter	Conditions	Min	Max	Units
SMC_1	MCK Rising to A2-A25 Valid	C_{ADD} = 0 pF		7.78	ns
		C_{ADD} derating		0.0336	ns/pF
SMC_2	MCK Rising to NBS0/A0 Valid	C_{NBS0} = 0 pF		5.21	ns
		C_{NBS0} derating		0.0336	ns/pF
SMC_3	MCK Rising to NBS1 Valid	C_{NBS1} = 0 pF		5.36	ns
		C_{NBS1} derating		0.0336	ns/pF
SMC_4	MCK Rising to A1/NBS2 Valid	C_{NBS2} = 0 pF		5.31	ns
		C_{NBS2} derating		0.0336	ns/pF
SMC_5	MCK Rising to NBS3 Valid	C_{NBS3} = 0 pF		5.2	ns
		C_{NBS3} derating		0.0336	ns/pF
SMC_6	MCK Rising to NCS Falling	C_{NCS} = 0 pF		4.95	ns
		C_{NCS} derating		0.0336	ns/pF
SMC_7	MCK Rising to NCS Rising	C_{NCS} = 0 pF		5.32	ns
		C_{NCS} derating		0.0336	ns/pF
SMC_8	MCK Rising to NRD Falling	C_{NRD} = 0 pF		4.90	ns
		C_{NRD} derating		0.0336	ns/pF
SMC_9	MCK Rising to NRD Rising	C_{NRD} = 0 pF		4.98	ns
		C_{NRD}derating		0.0336	ns/pF
SMC_{10}	MCK Rising to NWR Falling	C_{NWR} = 0 pF		5.31	ns
		C_{NWR} derating		0.0336	ns/pF
SMC_{11}	MCK Rising to NWR Rising	C_{NWR} = 0 pF		5.37	ns
		C_{NWR} derating		0.0336	ns/pF
SMC_{12}	NWAIT Minimum Pulse Width		2 x t_{CPMCK}		ns

Table 39-2. SMC Read Signals with Hold Settings

Symbol	Parameter	Conditions	Min	Max	Units
NRD Controlled (READ_MODE = 1)					
SMC_{13}	Data Setup before MCK Rising (2)		1.12		ns
SMC_{14}	Data Hold after MCK Rising (3)		-0.14		ns
SMC_{15}	Data Setup before NRD High (2)	C_{NRD} = 0 pF	6.1		ns
		C_{NRD}derating	0.0336		ns/pF
SMC_{16}	Data Hold after NRD High (3)	C_{NRD} = 0 pF	5.12		ns
		C_{NRD}derating	0.0336		ns/pF
SMC_{17}	NRD High to NBS0/A0 Change (1)	C_{NBS0} = 0 pF		nrd hold length * t_{CPMCK} + 0.2	ns
		C_{NBS0} derating		0.0336	ns/pF
SMC_{18}	NRD High to NBS1 Change (1)	C_{NBS1} = 0 pF		nrd hold length * t_{CPMCK} + 0.4	ns
		C_{NBS1} derating		0.0336	ns/pF
SMC_{19}	NRD High to NBS2/A1 Change (1)	C_{NBS2} = 0 pF		nrd hold length * t_{CPMCK} + 0.4	ns
		C_{NBS2} derating		0.0336	ns/pF
SMC_{20}	NRD High to NBS3 Change (1)	C_{NBS3} = 0 pF		nrd hold length * t_{CPMCK} + 0.2	ns
		C_{NBS3} derating		0.0336	ns/pF
SMC_{21}	NRD High to A2 - A25 Change (1)	C_{ADD} = 0 pF		nrd hold length * t_{CPMCK} + 2.8	ns
		C_{ADD} derating		0.0336	ns/pF
SMC_{22}	NRD High to NCS Inactive (1)	C_{NCS} = 0 pF		(nrd hold length - ncs rd hold length) * t_{CPMCK} + 0.4	ns
		C_{NCS} derating		0.0336	ns/pF
SMC_{23}	NRD Pulse Width	C_{NRD} = 0 pF	nrd pulse length * t_{CPMCK} + 0.1		ns
		C_{NRD} derating	0.0336		ns/pF
NCS Controlled (READ_MODE = 0)					
SMC_{24}	Data Setup before MCK Rising (4)		1.12		ns
SMC_{25}	Data Hold after MCK Rising (5)		-0.143		ns
SMC_{26}	Data Setup before NCS High (4)	C_{NCS} = 0 pF	6.44		ns
		C_{NCS} derating	0.0336		ns/pF
SMC_{27}	Data Hold after NCS High (5)	C_{NCS} = 0 pF	5.5		ns
		C_{NCS} derating	0.0336		ns/pF

Table 39-2. SMC Read Signals with Hold Settings

Symbol	Parameter	Conditions	Min	Max	Units
SMC_{28}	NCS High to NBS0/A0 Change [(1)]	C_{NBS0} = 0 pF		ncs rd hold length * t_{CPMCK} - 0.1	ns
		C_{NBS0} derating		0.0336	ns/pF
SMC_{29}	NCS High to NBS1 Change [(1)]	C_{NBS1} = 0 pF		ncs rd hold length * t_{CPMCK} + 0.1	ns
		C_{NBS1} derating		0.0336	ns/pF
SMC_{30}	NCS High to NBS2/A1 Change [(1)]	C_{NBS2} = 0 pF		ncs rd hold length * t_{CPMCK} + 0.1	ns
		C_{NBS2} derating		0.0336	ns/pF
SMC_{31}	NCS High to NBS3 Change [(1)]	C_{NBS3} = 0 pF		ncs rd hold length * t_{CPMCK} - 0.1	ns
		C_{NBS3} derating		0.0336	ns/pF
SMC_{32}	NCS High to A2 - A25 Change [(1)]	C_{ADD} = 0 pF		ncs rd hold length * t_{CPMCK} + 2.5	ns
		C_{ADD} derating		0.0336	ns/pF
SMC_{33}	NCS High to NRD Inactive [(1)]	C_{NCS} = 0 pF		(ncs rd hold length - nrd hold length)* t_{CPMCK} - 0.3	ns
		C_{NCS} derating		0.0336	ns/pF
SMC_{34}	NCS Pulse Width	C_{NCS} = 0 pF	ncs rd pulse length * t_{CPMCK} + 0.4		ns
		C_{NCS} derating	0.0336		ns/pF

Note: 1. hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “ncs rd hold length” or “nrd hold length”.
2. Only one of these two timings needs to be met.
3. Only one of these two timings needs to be met.
4. Only one of these two timings needs to be met.
5. Only one of these two timings needs to be met.

Table 39-3. SMC Read Signals with no Hold Settings

Symbol	Parameter	Conditions	Min	Max	Units
NRD Controlled (READ_MODE = 1)					
SMC_{35}	Data Setup before MCK Rising (1)		2.94		ns
SMC_{36}	Data Hold after MCK Rising (2)		-1.66		ns
SMC_{37}	Data Setup before NRD High (1)	C_{NRD} = 0 pF	7.92		ns
		C_{NRD}derating	0.0336		ns/pF
SMC_{38}	Data Hold after NRD High (2)	C_{NRD} = 0 pF	7.92		ns
		C_{NRD}derating	0.0336		ns/pF
NCS Controlled (READ_MODE = 0)					
SMC_{39}	Data Setup before MCK Rising (3)		1.12		ns
SMC_{40}	Data Hold after MCK Rising (4)		-0.143		ns
SMC_{41}	Data Setup before NCS High (3)	C_{NCS} = 0 pF	6.5		ns
		C_{NCS} derating	0.0336		ns/pF
SMC_{42}	Data Hold after NCS High (4)	C_{NCS} = 0 pF	5.5		ns
		C_{NCS} derating	0.0336		ns/pF

Notes:
1. Only one of these two timings needs to be met.
2. Only one of these two timings needs to be met.
3. Only one of these two timings needs to be met.
4. Only one of these two timings needs to be met.

Table 39-4. SMC Write Signals with Hold Settings

Symbol	Parameter	Conditions	Min	Max	Units
SMC_{43}	MCK Rising to Data Out Valid	C_{DATA} = 0 pF		5.38	ns
		C_{DATA} derating		0.0336	ns/pF
NWR Controlled (WRITE_MODE = 1)					
SMC_{44}	Data Out Valid before NWR High	C = 0 pF	(nwr pulse length - 1) * t_{CPMCK}		ns
		C_{DATA} derating	0.0336		ns/pF
		C_{NWR} derating	0.0336		ns/pF
SMC_{45}	Data Out Valid after NWR High (1)	C = 0 pF	nwr hold length * t_{CPMCK}		ns
		C_{DATA} derating	0.0336		ns/pF
		C_{NWR} derating	0.0336		ns/pF
SMC_{46}	NWR High to NBS0/A0 Change (1)	C_{NBS0} = 0 pF	nwr hold length * t_{CPMCK} - 0.2		ns
		C_{NBS0} derating	0.0336		ns/pF
SMC_{47}	NWR High to NBS1 Change (1)	C_{NBS1} = 0 pF	nwr hold length * t_{CPMCK}		ns
		C_{NBS1} derating	0.0336		ns/pF
SMC_{48}	NWR High to NBS2/A1 Change (1)	C_{NBS2} = 0 pF	nwr hold length * t_{CPMCK}		ns
		C_{NBS2} derating	0.0336		ns/pF
SMC_{49}	NWR High to NBS3 Change (1)	C_{NBS3} = 0 pF	nwr hold length * t_{CPMCK} - 0.2		ns
		C_{NBS3} derating	0.0336		ns/pF
SMC_{50}	NWR High to A2 - A25 Change (1)	C_{ADD} = 0 pF	nwr hold length * t_{CPMCK} + 2.4		ns
		C_{ADD} derating	0.0336		ns/pF
SMC_{51}	NWR High to NCS Inactive (1)	C_{NCS} = 0 pF	(nwr hold length - ncs wr hold length)* t_{CPMCK} - 0.1		ns
		C_{NCS} derating	0.0336		ns/pF
SMC_{52}	NWR Pulse Width	C_{NWR} = 0 pF	nwr pulse length * t_{CPMCK} - 0.1		ns
		C_{NWR} derating	0.0336		ns/pF
NCS Controlled (WRITE_MODE = 0)					
SMC_{53}	Data Out Valid before NCS High	C = 0 pF	(ncs wr pulse length - 1)* t_{CPMCK} - 0.1		ns
		C_{DATA} derating	0.0336		ns/pF
		C_{NCS} derating	0.0336		ns/pF



Table 39-4. SMC Write Signals with Hold Settings

Symbol	Parameter	Conditions	Min	Max	Units
SMC_{54}	Data Out Valid after NCS High [1]	C = 0 pF	ncs wr hold length * t_{CPMCK} + 0.1		ns
		C_{DATA} derating	0.0336		ns/pF
		C_{NCS} derating	0.0336		ns/pF
SMC_{55}	NCS High to NWR Inactive [1]	C_{NCS} = 0 pF	(ncs wr hold length - nwr hold length)* t_{CPMCK} + 0.1		ns
		C_{NCS} derating	0.0336		ns/pF

Note: 1. hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “ncs wr hold length” or “nwr hold length”.

Table 39-5. SMC Write Signals with no Hold Settings (NWR Controlled only)

Symbol	Parameter	Conditions	Min	Max	Units
SMC_{56}	NWR Rising to A2-A25 Valid	C_{ADD} = 0 pF	8.18		ns
		C_{ADD} derating	0.0336		ns/pF
SMC_{57}	NWR Rising to NBS0/A0 Valid	C_{NBS0} = 0 pF	4.87		ns
		C_{NBS0} derating	0.0336		ns/pF
SMC_{58}	NWR Rising to NBS1 Change	C_{NBS1} = 0 pF	4.66		ns
		C_{NBS1} derating	0.0336		ns/pF
SMC_{59}	NWR Rising to A1/NBS2 Change	C_{NBS2} = 0 pF	5.01		ns
		C_{NBS2} derating	0.0336		ns/pF
SMC_{60}	NWR Rising to NBS3 Change	C_{NBS3} = 0 pF	4.74		ns
		C_{NBS3} derating	0.0336		ns/pF
SMC_{61}	NWR Rising to NCS Rising	C_{NCS} = 0 pF	5.7		ns
		C_{NCS} derating	0.0336		ns/pF
SMC_{62}	Data Out Valid before NWR Rising	C = 0 pF	(nwr pulse length - 1) * t_{CPMCK}		ns
		C_{DATA} derating	0.0336		ns/pF
		C_{NWR} derating	0.0336		ns/pF
SMC_{63}	Data Out Valid after NWR Rising	C = 0 pF	6.59		ns
		C_{DATA} derating	0.0336		ns/pF
		C_{NWR} derating	0.0336		ns/pF
SMC_{64}	NWR Pulse Width	C_{NWR} = 0 pF	nwr pulse length * t_{CPMCK} + 0.1		ns
		C_{NWR} derating	0.0336		ns/pF

Table 39-6. SMC Read Signals in Page Mode

Symbol	Parameter	Conditions	Min	Max	Units
SMC_{65}	Data Setup before MCK Rising		1.12		ns
SMC_{66}	Data Hold after MCK Rising		-0.143		ns



Figure 39-4. SMC Signals for NCS Controlled Accesses

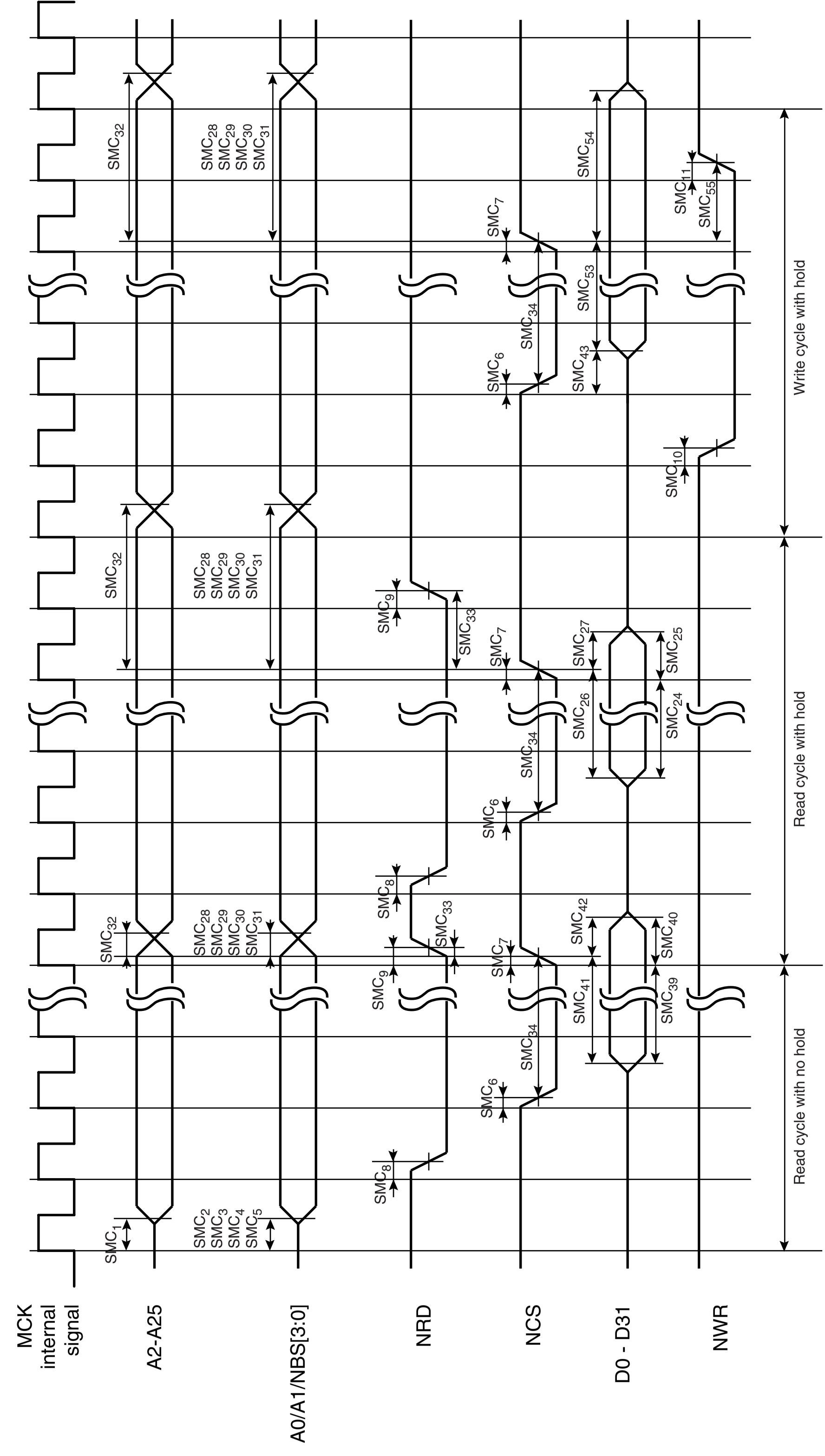


Figure 39-5. SMC Signals for NRD and NWR Controlled Accesses

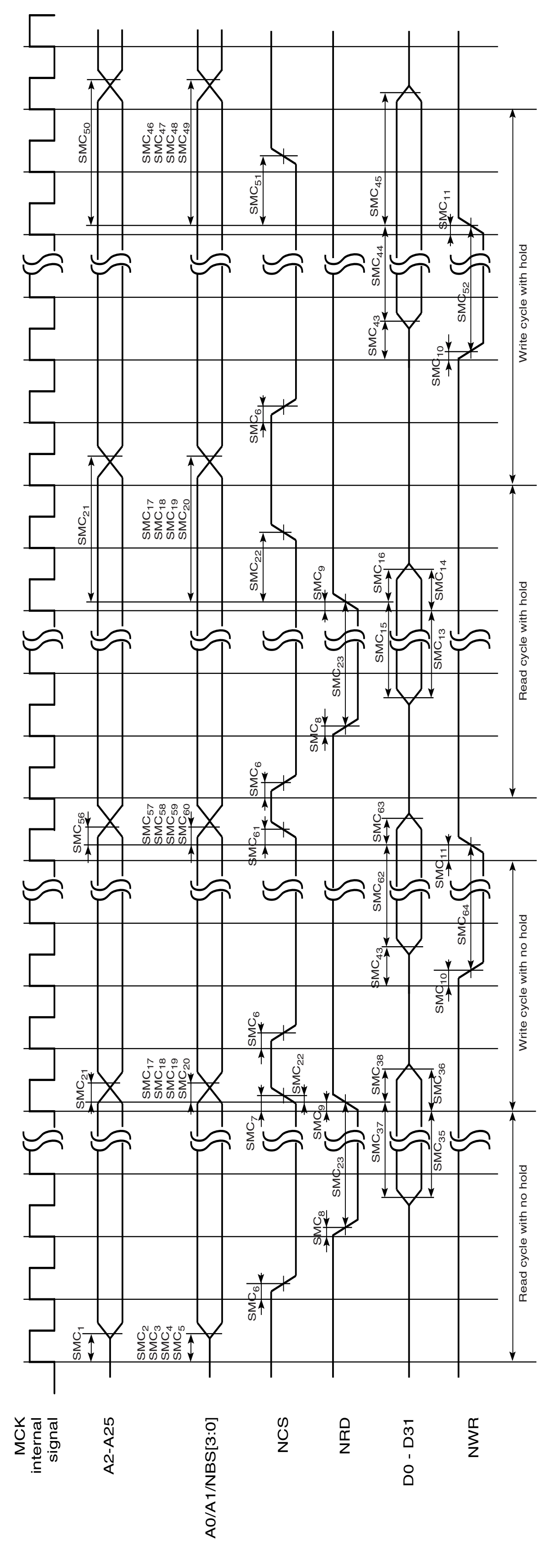
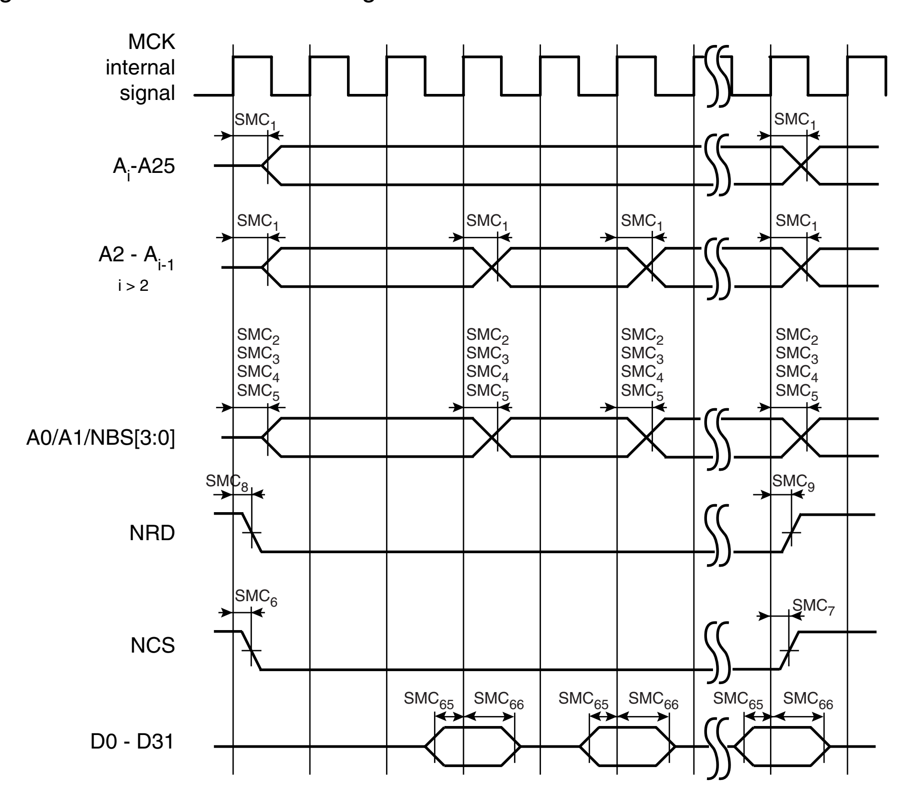




Figure 39-6. SMC Signals for Read Accesses in Page Mode [(1)]



Note: 1. $2 \leq i \leq 5$ and depends on page size. See SMC description for more information.

39.2.2 SDRAMC Signals Relative to SDCK

Table 39-7 and Table 39-8 below show timings relative to operating condition limits defined in the section "Conditions and Timings Computation" on page 618.

Table 39-7. SDRAMC Clock Signal

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPSDCK})$	SDRAM Controller Clock Frequency			119	MHz
t_{CPSDCK}	SDRAM Controller Clock Period		8.38		ns

Table 39-8. SDRAMC Signals

Symbol	Parameter	Conditions	Min	Max	Units
$SDRAMC_1$	SDCKE High before SDCK Rising Edge [(1)]	C_{SDCKE} = 0 pF	t_{CLMCK} + 4.62		ns
		C_{SDCKE} derating	0.0187		ns/pF
$SDRAMC_2$	SDCKE Low after SDCK Rising Edge [(1)]	C_{SDCKE} = 0 pF	t_{CHMCK} - 5.04		ns
		C_{SDCKE} derating	0.0167		ns/pF

Table 39-8. SDRAMC Signals

Symbol	Parameter	Conditions	Min	Max	Units
$SDRAMC_3$	SDCKE Low before SDCK Rising Edge [1]	C_{SDCKE} = 0 pF	t_{CLMCK} + 4.62		ns
		C_{SDCKE} derating	0.0187		ns/pF
$SDRAMC_4$	SDCKE High after SDCK Rising Edge [1]	C_{SDCKE} = 0 pF	t_{CHMCK} - 5.04		ns
		C_{SDCKE} derating	0.0167		ns/pF
$SDRAMC_5$	SDCS Low before SDCK Rising Edge [1]	C_{SDCS} = 0 pF	t_{CLMCK} + 4.57		ns
		C_{SDCS} derating	0.0336		ns/pF
$SDRAMC_6$	SDCS High after SDCK Rising Edge [1]	C_{SDCS} = 0 pF	t_{CHMCK} - 4.83		ns
		C_{SDCS} derating	0.0336		ns/pF
$SDRAMC_7$	RAS Low before SDCK Rising Edge [1]	C_{RAS} = 0 pF	t_{CLMCK} + 4.57		ns
		C_{RAS} derating	0.0336		ns/pF
$SDRAMC_8$	RAS High after SDCK Rising Edge [1]	C_{RAS} = 0 pF	t_{CHMCK} - 4.56		ns
		C_{RAS} derating	0.0167		ns/pF
$SDRAMC_9$	SDA10 Change before SDCK Rising Edge [1]	C_{SDA10} = 0 pF	t_{CLMCK} + 4.48		ns
		C_{SDA10} derating	0.0187		ns/pF
$SDRAMC_{10}$	SDA10 Change after SDCK Rising Edge [1]	C_{SDA10} = 0 pF	t_{CHMCK} - 4.67		ns
		C_{SDA10} derating	0.0167		ns/pF
$SDRAMC_{11}$	Address Change before SDCK Rising Edge [1]	C_{ADD} = 0 pF	t_{CLMCK} + 4.91		ns
		C_{ADD} derating	0.0336		ns/pF
$SDRAMC_{12}$	Address Change after SDCK Rising Edge [1]	C_{ADD} = 0 pF	t_{CHMCK} - 4.21		ns
		C_{ADD} derating	0.0336		ns/pF
$SDRAMC_{13}$	Bank Change before SDCK Rising Edge [1]	C_{BA} = 0 pF	t_{CLMCK} + 4.79		ns
		C_{BA} derating	0.0336		ns/pF
$SDRAMC_{14}$	Bank Change after SDCK Rising Edge [1]	C_{BA} = 0 pF	t_{CHMCK} - 4.32		ns
		C_{BA} derating	0.0336		ns/pF
$SDRAMC_{15}$	CAS Low before SDCK Rising Edge [1]	C_{CAS} = 0 pF	t_{CLMCK} + 4.68		ns
		C_{CAS} derating	0.0190		ns/pF
$SDRAMC_{16}$	CAS High after SDCK Rising Edge [1]	C_{CAS} = 0 pF	t_{CHMCK} - 5.08		ns
		C_{CAS} derating	0.0167		ns/pF
$SDRAMC_{17}$	DQM Change before SDCK Rising Edge [1]	C_{DQM} = 0 pF	t_{CLMCK} + 4.67		ns
		C_{DQM} derating	0.0336		ns/pF

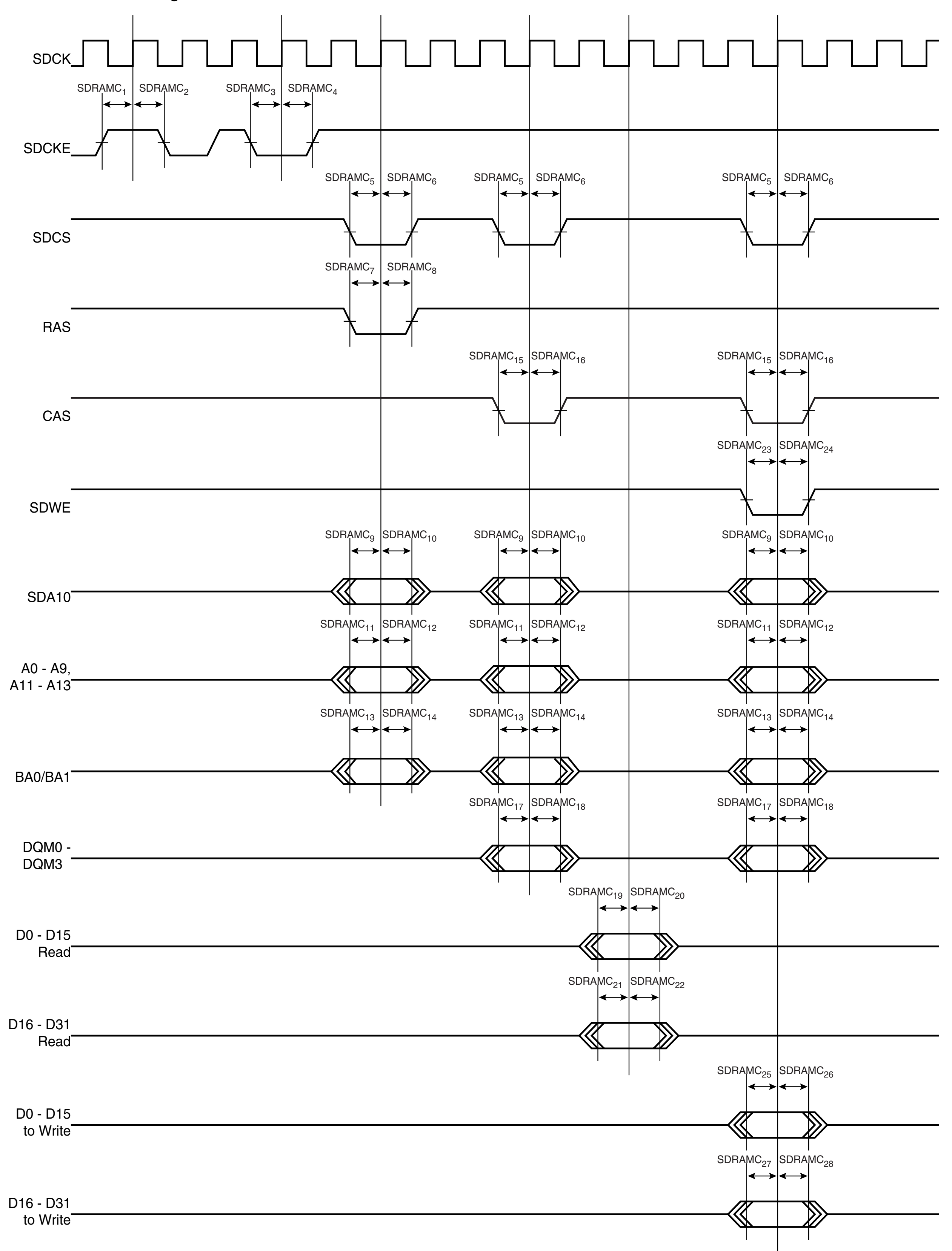


Table 39-8. SDRAMC Signals

Symbol	Parameter	Conditions	Min	Max	Units
$SDRAMC_{18}$	DQM Change after SDCK Rising Edge [1]	C_{DQM} = 0 pF	t_{CHMCK} - 4.38		ns
		C_{DQM} derating	0.0336		ns/pF
$SDRAMC_{19}$	D0-D15 in Setup before SDCK Rising Edge		1.95		ns
$SDRAMC_{20}$	D0-D15 in Hold after SDCK Rising Edge		-0.12		ns
$SDRAMC_{21}$	D16-D31 in Setup before SDCK Rising Edge		1.12		ns
$SDRAMC_{22}$	D16-D31 in Hold after SDCK Rising Edge		-0.19		ns
$SDRAMC_{23}$	SDWE Low before SDCK Rising Edge	C_{SDWE} = 0 pF	t_{CLMCK} + 4.47		ns
		C_{SDWE} derating	0.0187		ns/pF
$SDRAMC_{24}$	SDWE High after SDCK Rising Edge	C_{SDWE} = 0 pF	t_{CHMCK} - 4.61		ns
		C_{SDWE} derating	0.0167		ns/pF
$SDRAMC_{25}$	D0-D15 Out Valid before SDCK Rising Edge	C = 0 pF	t_{CLMCK} - 7.65		ns
		C_{DATA} derating	0.0336		ns/pF
$SDRAMC_{26}$	D0-D15 Out Valid after SDCK Rising Edge	C = 0 pF	t_{CHMCK} - 4.24		ns
		C_{DATA} derating	0.0336		ns/pF
$SDRAMC_{27}$	D16-D31 Out Valid before SDCK Rising Edge	C = 0 pF	t_{CLMCK} - 6.98		ns
		C_{DATA} derating	0.0336		ns/pF
$SDRAMC_{28}$	D16-D31 Out Valid after SDCK Rising Edge	C = 0 pF	t_{CHMCK} - 4.41		ns
		C_{DATA} derating	0.0336		ns/pF

Note: 1. The derating factor is not to be applied to t_{CLMCK} or t_{CHMCK}.

Figure 39-7. SDRAMC Signals Relative to SDCK



40. AT91SAM9261 Mechanical Characteristics

40.1 Thermal Considerations

40.1.1 Thermal Data

Table 40-1 summarizes the thermal resistance data depending on the package.

Table 40-1. Thermal Resistance Data

Symbol	Parameter	Condition	Package	Typ	Unit
θ_{JA}	Junction-to-ambient thermal resistance	Still Air	LFBGA217	39.1	°C/W
θ_{JC}	Junction-to-case thermal resistance		LFBGA217	7.1	

40.1.2 Junction Temperature

The average chip-junction temperature, T_J, in °C can be obtained from the following:

1. $T_J = T_A + (P_D \times \theta_{JA})$
2. $T_J = T_A + (P_D \times (\theta_{HEATSINK} + \theta_{JC}))$

where:

- θ_{JA} = package thermal resistance, Junction-to-ambient (°C/W), provided in Table 40-4 on page 635.
- θ_{JC} = package thermal resistance, Junction-to-case thermal resistance (°C/W), provided in Table 40-4 on page 635.
- $\theta_{HEAT\ SINK}$ = cooling device thermal resistance (°C/W), provided in the device datasheet.
- P_D = device power consumption (W) estimated from data provided in the section "Power Consumption" on page 612.
- T_A = ambient temperature (°C).

From the first equation, the user can derive the estimated lifetime of the chip and decide if a cooling device is necessary or not. If a cooling device is to be fitted on the chip, the second equation should be used to compute the resulting average chip-junction temperature T_J in °C.

40.2 Package Drawings

Figure 40-1. 217-balls LFBGA Package Drawing

Ball Pitch : 0.80	Substrate Thickness : 0.36
Ball Diameter : 0.40	Mold Thickness : 0.53

Table 40-2. Soldering Information

Ball Land	0.43 mm ± 0.05
Solder Mask Opening	0.30 mm ± 0.05

Table 40-3. Device and 217-ball LFBGA Package Maximum Weight

450	mg

Table 40-4. 217-ball LFBGA Package Characteristics

Moisture Sensitivity Level	3

Table 40-5. Package Reference

JEDEC Drawing Reference	MO-205
JESD97 Classification	e1

40.3 Soldering Profile

Table 40-6 gives the recommended soldering profile from J-STD-20.

Table 40-6. Soldering Profile

Profile Feature	Green Package
Average Ramp-up Rate (217°C to Peak)	3° C/sec. max.
Preheat Temperature 175°C ±25°C	180 sec. max.
Temperature Maintained Above 217°C	60 sec. to 150 sec.
Time within 5° C of Actual Peak Temperature	20 sec. to 40 sec.
Peak Temperature Range	260 +0 ° C
Ramp-down Rate	6° C/sec. max.
Time 25° C to Peak Temperature	8 min. max.

Note: It is recommended to apply a soldering temperature higher than 250°C.

A maximum of three reflow passes is allowed per component.

41. AT91SAM9261 Ordering Information

Table 41-1. AT91SAM9261 Ordering Information

Ordering Code	Package	Package Type	Temperature Operating Range
AT91SAM9261-CJ	BGA217	RoHS-compliant	Industrial -40°C to 85°C



42. AT91SAM9261 Errata

These errata refer to AT91SAM9261 devices packaged in 217-ball BGA with the marking AT91SAM9261-CJ.

42.1 Boot ROM

42.1.1 Watchdog Disable

When AT91SAM9261 boots on internal ROM (BMS =1) the watchdog timer is disabled by software. Because the watchdog mode register is a “write once”, the system, designed to boot on an SPI DataFlash, cannot reuse the watchdog functionality.

Problem Fix/Workaround

If the watchdog feature is mandatory for the application, the system must boot on external memory connected on CS0 (BMS = 0).

42.2 SDRAM Controller

42.2.1 SDCLK Clock active after reset

After a reset the SDRAM clock is always active leading in over consumption in the pad.

Problem Fix/Workaround

The following sequence allows to stop the SDRAM clock.

1. Set the bit LPCB in the SDRAMC Low Power Register.
2. Write 0 in the SDRAMC Mode Register and perform a dummy write in sdram to complete.

42.3 Serial Peripheral Interface (SPI)

42.3.1 Pulse Generation on SPCK

In Master Mode, there is an additional pulse generated on SPCK when the SPI is configured as follows:

– The Baudrate is odd and different from 1.

– The Polarity is set to 1.

– The Phase is set to 0 .

Problem Fix/Workaround

None.

42.3.2 Bad tx_ready behavior when CSAAT=1 and SCBR = 1

If the SPI2 is programmed with CSAAT = 1, SCBR(baudrate) = 1 and two transfers are performed consecutively on the same slave with an IDLE state between them, the tx_ready signal does not rise after the second data has been transferred in the shifter. This can imply for example that the second data is sent twice.

Problem Fix/Workaround

None. Do not use the combination CSAAT=1 and SCBR =1.

42.3.3 LASTXFER (Last Transfer) Behavior

In FIXED Mode, with CSAAT bit set, and in "PDC mode" the Chip Select can rise depending on the data written in the SPI_TDR when the TX_EMPTY flag is set. For example, if the PDC writes a "1" in bit 24 (LASTXFER bit) of the SPI_TDR, the Chip Select rises as soon as the TXEMPTY flag is set.

Problem Fix/Workaround

Use the CS in PIO mode when "PDC mode" is required and CS has to be maintained between transfers.

42.4 Serial Synchronous Controller (SSC)

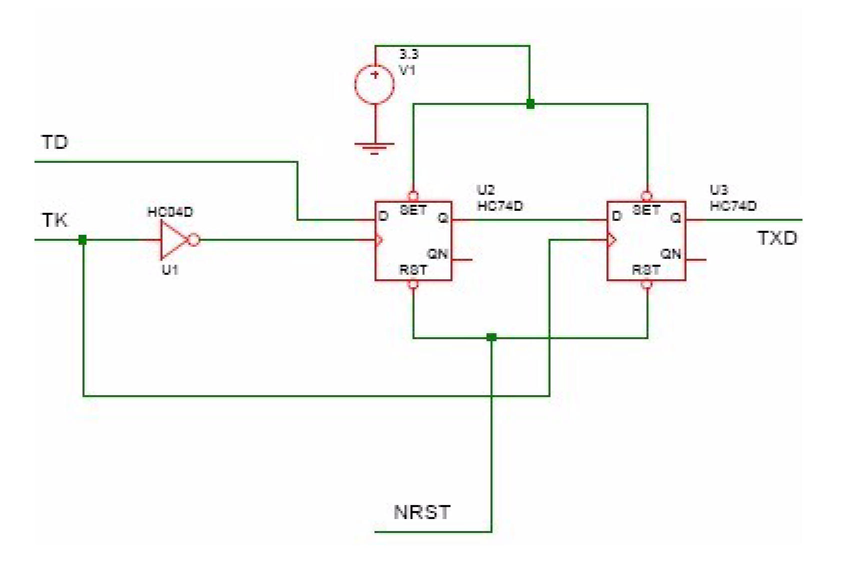
42.4.1 Transmitter Limitations in Slave Mode

If TK are programmed as input and TF is programmed as output and requested to be set to low/high during data emission, the Frame Synchro is generated one bit clock period after the data start, one data bit is lost. This problem does not exist when generating periodic synchro.

Problem Fix/Workaround

The data need to be delayed for one bit clock period with an external assembly.

In the following schematic, TD, TK and NRST are AT91SAM9261 signals, TXD is the delayed data to connect to the device.



42.4.2 Transmitter Limitations in Slave Mode

If TK is programmed as output and TF is programmed as input, it is impossible to emit data when start of edge (rising or falling) of synchro with a Start Delay equal to zero.

Problem Fix/Workaround

None.

42.4.3 Periodic Transmission Limitations in Master Mode

If Last Significant Bit is sent first (MSBF = 0) the first TAG during the frame synchro is not sent.

Problem Fix/Workaround

None.

42.5 Two-wire Interface (TWI)

42.5.1 Clock Divider

The value of CLDIV x 2^{CKDIV} must be less than or equal to 8191, the value of CHDIV x 2^{CKDIV} must be less than or equal to 8191·

Problem Fix/Workaround

None.

42.5.2 Behavior of OVRE Bit

In Master Mode during a read access, if the sequence described as follows occurs:

1. A byte is received but not read through the TWI_RHR.
2. A step command is performed through the TWI_CR to end the read access.
3. The last data byte is received.

The Overrun Flag (OVRE) does not rise, whereas a data byte has been lost.

Problem Fix/Workaround

None.

42.6 USART

42.6.1 Hardware Handshake

The Hardware Handshake does not work at speeds higher than 750 kbauds.

Problem Fix/Workaround

None.

43. Revision History

Table 43-1. Revision History

Doc. Rev.	Date	Comments	Change Request Ref.
6062A	02-Jun-05	First issue.	

Table 43-1. Revision History (Continued)

Doc. Rev.	Date	Comments	Change Request Ref.
6062B	14-Oct-05	Changed SPI pin names in Figure 2-1, "AT91SAM9261 Block Diagram," on page 4, Table 3-1, "Signal Description by Peripheral," on page 5, Table 10-1, "Multiplexing on PIO Controller A," on page 30, Table 10-2, "Multiplexing on PIO Controller B," on page 31 and Table 10-3, "Multiplexing on PIO Controller C," on page 32.	05-398
		Corrected EBI/Compact Flash interface description with updated A22 pin functionality in Figure 2-1, "AT91SAM9261 Block Diagram," on page 4, Figure 20-1, "Organization of the External Bus Interface," on page 131 and Table 20-6, "CFCE1 and CFCE2 Truth Table," on page 139.	05-481
		Changed value of programmable pull-up resistor in Section 6.4 "PIO Controller A, B and C Lines" on page 12.	05-496
		Bus Matrix: Corrected reset values of Slave Configuration Registers 0, 1 and 2 in Table 19-1, "Register Mapping," on page 124.	05-498
		Bus Matrix: Updated bit information in Section 19.5.5 "USB Pad Pull-up Control Register" on page 129.	05-486
		SMC: Corrected reset values of SMC_SETUP, SMC_CYCLE and SMC_MODE in Table 21-9, "SMC Register Mapping," on page 181.	05-499
		SDRAMC: Removed Hardware Interface section from Section 22.4 "Application Example" on page 188.	05-468
		SDRAMC: Updated Figure 22-2, "SDRAM Device Initialization Sequence," on page 191.	05-479
		PMC: Updated Section 25.8 "Programming Sequence" on page 229.	05-393
		PMC: Updated Figure 25-5, "Change PLLA Programming," on page 235 and added Figure 25-6, "Change PLLB Programming," on page 235.	05-198
		PMC: Added important note on programming Bit 29 of CKGR_PLLAR in Section 25.8 "Programming Sequence" on page 229 and in Section 25.10.9 "PMC Clock Generator PLL A Register" on page 245.	05-239
		AIC: Added information on external interrupt sources for bit SRCTYPE in Section 26.8.3 "AIC Source Mode Register" on page 267.	05-269
		DBGU: Updated bit description for SRAMSIZ in Section 27.5.10 "Debug Unit Chip ID Register" on page 300.	05-306
		PIO: Removed reference to resistor value in Section 28.4.1 "Pull-up Resistor Control" on page 311.	05-497
		SPI: References to MCK/32 removed throughout. Figure 29-1, "Block Diagram," on page 338 and Figure 29-5, "Master Mode Block Diagram," on page 343 changed.	05-484
		SPI: Section 29.7.5 "SPI Status Register" on page 356 SPI_RCR, SPI_RNCR, SPI_TCR, SPI_TNCR location defined.	04-183
		SPI: Section 29.7.4 "SPI Transmit Data Register" on page 355, LASTXFER: Last Transfer text added.	05-434
		SPI: Section 29.7.2 "SPI Mode Register" on page 352, PCSDEC: Chip Select Decode on changed	05-476
		MCI: Corrected pin names in Figure 34-4, "MMC Bus Connections (One Slot)," on page 498.	05-308
		UHP: Added information on memory access errors in "USB Host Port (UHP)" on page 522.	05-240





Table 43-1. Revision History (Continued)

Doc. Rev.	Date	Comments	Change Request Ref.
6062B	30-Aug-05	Updated R_{PULLUP} values in Table 38-2, "DC Characteristics," on page 611.	Review
		Corrected backup consumption value in Table 38-3, "Power Consumption for Different Modes(1)," on page 613.	05-332
		Removed data on USB Transceiver Switching Characteristics at low speed in Section 38.6 "USB Transceiver Characteristics" on page 616.	Review
		Corrected value of R_{EXT} in Figure 38-2, "USB Data Signal Rise and Fall Times," on page 617.	Review
		Removed MTBF data from Section 40.1 "Thermal Considerations" on page 634.	05-345
		Inserted Table 40-2, "Soldering Information," on page 635.	05-347
		Updated Section 41. "AT91SAM9261 Ordering Information" on page 637.	05-485
		Added Section 42. "AT91SAM9261 Errata" on page 638.	



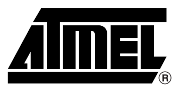


Table of Contents







Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature

ARM POWERED®



6062B–ATARM–15-Nov-05